UNIVERSITY OF CALIFORNIA

Los Angeles

Variational and PDE-based methods for big data analysis, classification and image processing using graphs

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Ekaterina Merkurjev

2015

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE
2015

2. REPORT TYPE

3. DATES COVERED
00-00-2015 to 00-00-2015

4. TITLE AND SUBTITLE
Variational and PDE-based Methods for Big Data Analysis, Classification and Image Processing Using Graphs

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California, Los Angeles,Department of Mathematics,Los Angeles,CA,90095

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present several graph-based algorithms for image processing and classification of highdimensional data. The first (semi-supervised) method uses a graph adaptation of the classical numerical Merriman-Bence-Osher (MBO) scheme, and can be extended to the multiclass case via the Gibbs simplex. We show examples of the application of the algorithm in the areas of image inpainting (both binary and grayscale), image segmentation and classification on benchmark data sets. We have also applied this algorithm to the problem of object detection using hyperspectral video sequences as a data set. In addition, a second related model is introduced. It uses a diffuse interface model based on the Ginzburg-Landau functional, related to total variation compressed sensing and image processing. A multiclass extension is introduced using the Gibbs simplex with the functional???s double-well potential modified to handle the multiclass case. The version minimizes the functional using a convex splitting numerical scheme. In our computations, we make use of fast numerical solvers for finding the eigenvectors and eigenvalues of the graph Laplacian and take advantage of the sparsity of the matrix. Experiments on benchmark data sets show that our models produce results that are comparable with or outperform the state-of-the-art algorithms. The second (semi-supervised) method develops a global minimization framework for binary classification of high-dimensional data. It combines recent convex optimization methods for image processing with recent graph based variational models for data segmentation. Two convex splitting algorithms are proposed, where graph-based PDE techniques are used to solve some of the subproblems. It is shown that global minimizers can be guaranteed for semi-supervised segmentation with two regions. If constraints on the volume of the regions are incorporated, global minimizers cannot be guaranteed, but can often be obtained in practice and otherwise be closely approximated. We perform a thorough comparison to recent MBO (Merriman-Bence-Osher) [81] and phase field methods, and show the advantage of the proposed algorithms. Lastly, we present the current work (unsupervised method) related to normalized cuts. The method uses a clever alternative to the normalized cut to solve the binary classification problem. In particular, we work with the Ginzburg-Landau functional. In addition, we use a generalized graphical framework, so several different graph Laplacians are tested and their results are compared.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**132**	

ABSTRACT OF THE DISSERTATION

Variational and PDE-based methods for big data analysis, classification and image processing using graphs

by

Ekaterina Merkurjev

Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2015
Professor Andrea Bertozzi, Chair

We present several graph-based algorithms for image processing and classification of high-dimensional data. The first (semi-supervised) method uses a graph adaptation of the classical numerical Merriman-Bence-Osher (MBO) scheme, and can be extended to the multiclass case via the Gibbs simplex. We show examples of the application of the algorithm in the areas of image inpainting (both binary and grayscale), image segmentation and classification on benchmark data sets. We have also applied this algorithm to the problem of object detection using hyperspectral video sequences as a data set. In addition, a second related model is introduced. It uses a diffuse interface model based on the Ginzburg-Landau functional, related to total variation compressed sensing and image processing. A multiclass extension is introduced using the Gibbs simplex, with the functional's double-well potential modified to handle the multiclass case. The version minimizes the functional using a convex splitting numerical scheme. In our computations, we make use of fast numerical solvers for finding the eigenvectors and eigenvalues of the graph Laplacian, and take advantage of the sparsity of the matrix. Experiments on benchmark data sets show that our models produce results that are comparable with or outperform the state-of-the-art algorithms.

The second (semi-supervised) method develops a global minimization framework for binary classification of high-dimensional data. It combines recent convex optimization methods for image processing with recent graph based variational models for data segmentation. Two convex splitting algorithms are proposed, where graph-based PDE techniques are used to solve some of the subproblems. It is shown that global minimizers can be guaranteed for semi-supervised segmentation with two regions. If constraints on the volume of the regions are incorporated, global minimizers cannot be guaranteed, but can often be obtained in practice and otherwise be closely approximated. We perform a thorough comparison to recent MBO (Merriman-Bence-Osher) [81] and phase field methods, and show the advantage of the proposed algorithms.

Lastly, we present the current work (unsupervised method) related to normalized cuts. The method uses a clever alternative to the normalized cut to solve the binary classification problem. In particular, we work with the Ginzburg-Landau functional. In addition, we use a generalized graphical framework, so several different graph Laplacians are tested and their results are compared.

The dissertation of Ekaterina Merkurjev is approved.

Chris Anderson

Mark Cohen

Stanley Osher

Andrea Bertozzi, Committee Chair

University of California, Los Angeles

2015

To my family and friends

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to thank my advisor Andrea Bertozzi. She has provided me with invaluable advice, and guided me in all aspects of my graduate student career. Without her support and collaboration, this dissertation would not exist. I would also like to thank Stanley Osher, Chris Anderson and Mark Cohen for being part of my dissertation committee and for their time.

Second, I would like to thank all my collaborators. I would like to thank Tijana Kostic for being one of the coauthors of my very first paper in graduate school. I am also grateful for having worked with Torin Gerhart, Justin Sunu, Lauren Liu, J.-M. Chang and Jerome Gilles as part of a research experience for undergraduates out of which a great paper was produced. I would also like to thank Cristina Garcia, Allon Percus and Arjuna Flenner for collaborating with me on more than one paper. Moreover, I would like to thank Egil Bae and X.-C. Tai for their insight and suggestions on our recent paper. Moreover, much thanks to Huiyi, Yi, etc. for useful discussions.

The last five years of my life would not have been the same if I did not have the support of my wonderful family and friends. I would like to thank Michelle, Zagid, Andreea, Alexey and my other friends for making my time in graduate school very memorable. I would like to thank my father, Alexander, for being the best dad ever and for putting up with my math questions. Many thanks to my amazing and wonderful mom, Olga, for putting up with me in all non-mathematical areas of life, and my sister, Daria, for being the most loving sister in the world and great company! Last, I would like to thank Igor for his love and support, and for just being an amazing man!

The work in this dissertation was supported by NSF grants DMS-1118971, DMS-1417674 and DMS-0914856, ONR grants N000141210040, N0001413WX20136 and N00014120838, the AFOSR MURI grant FA9550-10-1-0569, and the W. M. Keck Foundation. I was also supported by the Eugene Cota-Robles fellowship, the NSF Fellowship and the Dissertation Year Fellowship during the PhD program.

VITA

Education

University of California, Los Angeles,

Ph.D., Candidate, Applied Mathematics, **expected June 2015**

- *GPA: 4.00*

B.S./M.S. (Joint Program), Applied Mathematics **June 2010**

- *GPA: 3.954*

Expertise

- Background in applied and computational mathematics, optimization, numerical analysis, algorithms, differential equations, scientific computing.
- Programming skills: C++, Python, Matlab, OpenMP, UNIX.

Awards

- UC President's Postdoctoral Fellowship (2015-2017)
- Pacific Journal of Mathematics Dissertation Prize (2015)
- Dissertation Year Fellowship (2014-2015)
- NSF Graduate Fellowship (2011-2014)
- Eugene-Cota Robles Fellowship (2010-2011)
- NSF Research and Training Grant (RTG) Fellowship in Applied Mathematics (2010-2011)
- Sherwood Award (for excellence in undergraduate studies) (2010)
- Departmental Scholar at UCLA (2009-2010)
- Basil Gordon Prize ($1000) for Putnam exam (2008)

Journal/ Conference Publications

1. Merkurjev, E., Bae, E., Bertozzi, A.L., and Tai, X.-C. *Global Binary Data Optimization on Graphs for Data Segmentation*, to appear in J. Math. Imag. Vis.

2. Merkurjev, E., Sunu, J. and Bertozzi, A.L. *Graph MBO Method for Multiclass Segmentation of Hyperspectral Stand-off Detection Video*, IEEE International Conference on Image Processing, Paris, France, October 27-30, 2014.

3. Merkurjev, E., Garcia-Cardona, C., Bertozzi, A.L., Flenner, A. and Percus, A., *Research Announcement: Diffuse Interface Methods for Multiclass Segmentation of High-Dimensional Data*, Applied Mathematics Letters, 33, pp. 29-34, 2014.

4. Garcia-Cardona, C., Merkurjev, E., Bertozzi, A.L., Percus, A., Flenner, A. *Multiclass Segmentation Using the Ginzburg-Landau Functional and the MBO Scheme*, IEEE Trans. Pattern Anal. Mach. Intell., 36(8), pp. 1600-1614, 2014.

5. Gerhart, T., Sunu, J., Lieu, L., Merkurjev, E., Chang, J.-M., Gilles, J., Bertozzi, A.L. *Detection and Tracking of Gas Plumes in LWIR Hyperspectral Video Sequence Data*, SPIE Conference on Defense Security and Sensing, Baltimore, April 29-May 3, 2013.

6. Merkurjev, E., Kostić, T. and Bertozzi, A.L. *MBO Scheme on Graphs for Segmentation and Image Processing*, SIAM J. Imag. Sci., 6(4), pp. 1903-1930, 2013.

7. Peterson, G.E., Campbell, E.T., Balbas, J., Ivy, S., Merkurjev, E., Rodriguez, P., "*Relative Performance of Lambert Solvers 1: 0-Revolution Methods*, Adv Astronaut Sci, 136 (1), pg. 1495-1510, presented at 20th AAS/AIAA Space Flight Mechanics Meeting, San Diego, CA, February 14-17, 2010.

Conference Presentations/ Posters

- AWM Research Symposium, College Park, MD, April 11-12, 2015
- IEEE International Conference on Image Processing, Paris, Oct. 27-30, 2014
- Participant at Semidefinite Programming and Graph Algorithms Workshop, Providence, RI, Feb. 1- Mar. 1, 2014
- Algorithms for Threat Detection Workshop, Boulder, CO, March 10-12, 2014
- Fall Western Sectional Meeting (#1095), UCR, Riverside, CA, Nov. 2-3, 2013
- ONR Math Data Science Program Review Meeting, Durham, NC, Sept. 16-19, 2013
- Algorithms for Threat Detection Workshop, San Diego, CA, Nov. 26-29, 2012

Teaching/ Mentoring Experience

- Instructor of 2014 UCLA GRE Workshop (summer 2014)
- Mentor for RIPS program at Institute for Pure and Applied Mathematics (summer 2014)
- Mentor for Applied Mathematics REU at UCLA (summer 2012)
- Teaching Assistant for Calculus (winter 2011)

CHAPTER 1

Introduction

We present several methods, outlined in Chapters 3-5, for image processing and data classification using a graphical framework. The framework is often used to exploit underlying similarities in the data [6, 22, 106, 113–115]. For example, spectral graph theory [24, 84] uses this approach to perform various tasks in imaging and data clustering. Graph-based formulations have been also used extensively for image processing applications [10, 26, 27, 36, 56, 57, 59, 74, 96]. Specifically, algorithms for image denoising in [17], image inpainting and reconstruction in [51, 91, 111], image deblurring in [77] and manifold processing in [36] all utilize such formulations. We use a non-local calculus formulation [103] to generalize the continuous formulation to a (non-local) discrete setting, while other non-local versions for weighted graphs are described in [36]. A comprehensive reference about casting continuous PDEs in graph form is found in [58].

Chapter 3 develops a fast algorithm (MBO method) for classification and image processing. The method is inspired by diffuse interface models that have been used in a variety of problems, such as those in fluid dynamics and materials science. As an alternative to L^1 compressed sensing methods, Bertozzi and Flenner introduce a graph-based model based on the Ginzburg-Landau functional in their work [9]. To define the functional on a graph, the spatial gradient is replaced by a more general graph gradient operator. Analogous to the continuous case, the first variation of the model yields a gradient descent equation with the graph Laplacian, which is then solved by a numerical scheme with convex splitting. To reduce the dimension of the graph Laplacian and make the computation more efficient, the authors propose the Nyström extension method [44] to approximate eigenvalues and the corresponding eigenvectors of the graph Laplacian. Moreover, many applications suggest that the MBO scheme of Merriman, Bence and Osher [82] for approximating the motion by mean curvature performs very well in minimizing functionals built around the Ginzburg-Landau functional. For example, the authors of [39] propose an adaptation of the scheme to solve the piecewise constant Mumford-Shah functional. This inspired us to adapt the MBO scheme [82] for solving graph based equations to create a simple algorithm that achieves faster convergence through a small number of computationally inexpensive iterations. The resulting MBO method can be applied to various problems, such as image processing, classification and object detection in, for example, hyperspectral data.

We proceed with the chapter by presenting two graph-based algorithms for multiclass classification of high dimensional data. The multiclass extension is obtained using the Gibbs simplex. The first algorithm minimizes the Ginzburg-Landau functional using gradient descent and a convex-splitting scheme. The second algorithm is an extension of the MBO method already introduced. It uses fewer parameters than the first algorithm, and while this may in some cases make it more restrictive, in our experiments it was highly accurate and efficient. Both of these algorithms demonstrate how methods motivated by the PDE literature can be productively adapted to graphs, producing effective multiclass data segmentation methods.

The theoretical portion of the chapter is concluded with a presentation of an application of the multiclass MBO algorithm to hyperspectral video data. We use the Nyström extension method to efficiently calculate the needed eigenvectors. This implementation of the algorithm requires an operator assisted spectral clustering preprocessing step to identify a subset of pixels denoted as "ground truth" for the four classes. The resulting classification of chemical plumes and background pixels are excellent. Only a small number of eigenvectors is needed to achieve a good result and no preprocessing is necessary.

Chapter 4 develops a global minimization framework for segmentation of high dimensional data into two classes. Instead of applying classical combinatorial algorithms, we build on more recent work from imaging, which formulates two class partition problems as convex variational problems [11, 21, 54] or variational min-cut/max-flow problems [108, 109]. Convex optimization algorithms were used in [11, 54, 108, 109] to split the problems into simpler subproblems, each of which could be solved by PDE techniques. In this chapter, we describe the extension of the variational min-cut/max-flow duality in [108, 109] and of the algorithm in [54, 107] to a more general graph setting to solve a more general clustering problem. The new subproblems are solved by graph-based PDE techniques. We also show how constraints on the size of each class can be incorporated by a small modification of the max-flow algorithm. The advantage of the methods proposed in this chapter is the fact that they are convex, unlike those in the earlier chapter, which have the potential of occasionally being stuck in a local minima.

While Chapters 3 and 4 involve semi-supervised algorithms, *Chapter 5* develops an unsupervised method, where there is no a priori knowledge of the labeling of some of the data points. This is a harder problem, but it has the advantage of not requiring one to know part of the ground truth. The goal of the chapter is binary classification, same as in Chapter 4. The novel binary unsupervised clustering algorithm introduced is a modification of the normalized cut problem.

One of the ways to cluster a target set X (defined on a graph $G = (V, E)$ and the weight function w defined on the set of edges) into two clusters is to find a partition $X = S \cup \bar{S}$ such that the following value is minimized:

$$cut(S, \bar{S}) = \sum_{x \in S, y \in \bar{S}} w(x, y).$$

The intuition behind this is the fact that we want to find a partition such that the weights between vertices of the same set are large, and weights between vertices of different sets are small. In other words, we want to group vertices that are alike together, and put those that are dissimilar in different groups. However, minimizing the above problem usually leads to an undesirable solution, because it tends to isolate individual vertices from the rest of the graph. Usually, what is wanted is for the two sets to be relatively close in size. Thus, some sort of normalization is usually needed. Our method is developed from the normalized cut problem and uses the Ginzburg-Landau functional.

This problem has been a popular one to be studied. Here, we emphasize work that at least indirectly deals with solving the cut problem (with some normalization factor). In [18], the authors present a generalized version of spectral clustering using the graph p-Laplacian. They show that, in a certain limit, the cut resulting from thresholding the second eigenvector of the graph p-Laplacian is the solution to the Cheeger cut problem. An efficient scheme to calculate the eigenvector is introduced. In [63], Hein et al. show that some constrained optimization problems can be formulated as nonlinear eigenproblems. The authors then describe a generalization of the inverse power method which converges to nonlinear eigenvectors. This method is applied to spectral clustering and sparse principal component analysis. Recent work by Bresson, Szlam, Laurent, von Brecht, et al. includes several important papers related to this area. In [100], Szlam et al. give a continuous relaxation of the Cheeger cut problem on a weighted graph. In particular, they show the equivalence/relationship between the total variation problem and the Cheeger cut problem. An algorithm based on the split-Bregman method [55] is developed to minimize the proposed energy. Authors of [12] present two algorithms solving the relaxed Cheeger cut problem. They also prove convergence results for these algorithms. The first algorithm is a novel steepest descent approach and the second one is a modified inverse power method. In [14], Bresson et al. describe an adaptive version of the method shown in [12], the goal of which is to compute the solution of the relaxed Cheeger cut problem. This is achieved via a new adaptive stopping condition. The result is an algorithm that is monotonic and much more efficient than before. Multiclass extensions have also been proposed. One approach is to use recursive extensions and thus solve a collection of binary problems. However, other approaches have been introduced. The authors of [13] present a framework for multi class total variation clustering that does not use recursion. They formulate the Cheeger energy in a multi class setting, and the relax the energy in a continuous setting. This results in an optimization problem involving total variation, which is then solved using the proposed proximal splitting algorithm. In [15], an extension of the result of [100] is introduced. The extension deals with multiple classes and learning from these classes using a set of labels. The method is made even more efficient by the usage of fast L^1 solvers, designed for the total variation semi-norm.

CHAPTER 2

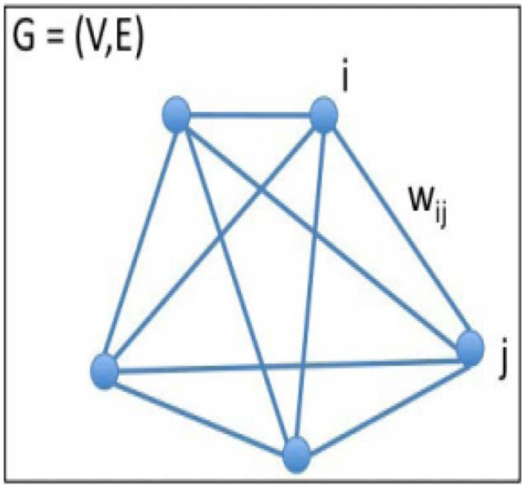
Background

2.1 Graphical Framework

For all methods, we consider a graphical framework with an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of vertices and $\mathbf{E}$ is the set of edges. Let $\mathbf{w}$ be the weight function (defined on the set of edges) which measures the similarity between each two vertices. Also, let

$$d(x) = \sum_{y \in V} w(x, y)$$

be the degree of vertex x. The diagonal matrix $\mathbf{D}$ contains the degree along its diagonal entries and the matrix $\mathbf{W}$ contains values of the weight function. Both matrices are of dimension n by n, where n is the number of vertices. A representation of the graphical framework is shown below.



One advantage of using a graphical framework is that it allows one to be non-local and take into account the relationship between any two nodes in the data set. Therefore, repetitive structure and texture can be captured. The graphical framework is also more general, and can be easily constructed for any data set.

Weight Function

When choosing a weight function, the goal is to give a large weight to an edge if the two vertices it is connecting are similar and a small weight if they are dissimilar. One popular choice for the weight function is the Gaussian

$$w(x, y) = e^{-\frac{d(x,y)^2}{\sigma^2}}, \tag{1}$$

where $d(x, y)$ is some distance measure between the two vertices x and y, and σ is a parameter to be chosen. For example, if the data set consists of points in $\mathbb{R}^2$, $d(x, y)$ can be the Euclidean distance between point x and point y, since points farther away are less likely to belong to the same cluster than points closer together. For images, $d(x, y)$ can be defined as the weighted 2-norm of the difference of the feature vectors of pixels x and y, where the feature vector of a node consists of intensity values of pixels in its neighborhood, as described in [52].

Another choice for the similarity function used in this work is the Zelnik-Manor and Perona weight function [90] for sparse matrices:

$$w(x, y) = e^{-\frac{d(x,y)^2}{\sqrt{\tau(x)\tau(y)}}}, \tag{2}$$

where the local parameter $\tau(x) = d(x, z)^2$, and z is the M^{th} closest vertex to vertex x.

Note that it is not necessary to use a *fully connected graph setting*, which might be a computational burden. Specifically, the fully connected graph can be approximated by a much smaller graph by only including an edge between vertex x and y if x is a k-nearest neighbor of y or vice versa. This is called a *k-nearest neighbor graph.* One can also create a *mutual k-nearest neighbor graph* by only including an edge between x and y if both of them are k-nearest neighbors of each other. In this paper, we make use of such an approximation; our edge set includes only edges between vertices that are near to each other. If two vertices x and y are not connected by an edge, then the weight between them is set to 0.

Graph Laplacian

In the graphical framework, it is possible to introduce some common mathematical operators in a graphical setting. For this section, we will only be concerned with the graph version of the differential Laplace operator. Although many versions exist, we mention the following three matrices that are related to the differential Δ operator:

- $\mathbf{L} = \mathbf{D} - \mathbf{W}$, unnormalized Laplacian
- $\mathbf{L}_s = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$, symmetric Laplacian
- $\mathbf{L_{rw}} = \mathbf{D^{-1}L}$, random walk Laplacian

The last two matrices represent normalized versions of the original Laplacian, as it is sometimes desirable to scale, especially in high dimensions. Note that we have the following equations:

$$\mathbf{L}u(x) = \sum_y w(x,y)(u(x) - u(y)),$$

$$\mathbf{L}_s u(x) = \frac{1}{d(x)} \sum_y w(x,y)(u(x) - u(y)),$$

$$\mathbf{L_{rw}} u(x) = \frac{1}{\sqrt{d(x)}} \sum_y w(x,y)\big(\frac{u(x)}{\sqrt{d(x)}} - \frac{u(y)}{\sqrt{d(y)}}\big).$$

The graph Laplacian $\mathbf{L}$ has the following easily shown properties:
1) $\mathbf{L}$ is symmetric and positive semi-definite.
2) $\mathbf{L}$ has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_n$.
3) The smallest eigenvalue of $\mathbf{L}$ is 0; eigenvector is just a constant vector.

The graph Laplacians $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ have the following easily shown properties:
1) $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ are positive semi-definite.
2) $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ have n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_n$.
3) λ is an eigenvalue of $\mathbf{L_{rw}}$ with eigenvector u if and only if λ is an eigenvalue of $\mathbf{L}_s$ with eigenvector $w = D^{\frac{1}{2}}u$.
4) The smallest eigenvalue of $\mathbf{L}_s$ and $\mathbf{L_{rw}}$ is 0.

It is also worthwhile to mention that the multiplicity of eigenvalue 0 equals the number of connected components in the graph.

Other Graph Operators

Another important operator that arises from the need to define variational methods on graphs is the mean curvature on graphs. This non-local operator was introduced by Osher and Shen in [89], who defined it via graph based p-Laplacian operators. p-Laplace operators are a family of quasilinear elliptic partial differential operators defined for $1 \leq p < \infty$:

$$\mathbf{L}^p(u) = \nabla \cdot (\mid \nabla u \mid^{p-2} \nabla u)$$

In the special case $p = 2$, the p-Laplacian is just a regular Laplacian. For $p = 1$, the p-Laplacian represents curvature.

The discrete graph version of p-Laplace operators is defined in [36] as:

$$\mathbf{L}^p(u(x)) = \frac{1}{p} \sum_{(x,y)\in E} w(x,y)(\|\nabla u(x)\|^{p-2} + \|\nabla u(y)\|^{p-2})(u(x) - u(y)).$$

Note that the graph 2-Laplacian is just the graph Laplacian, which is consistent with the continuous case.

Let us now define the mean curvature on graphs- the discrete analog of the mean curvature of the level curve of a function defined on a continuous domain of $\mathbb{R}^N$:

$$\kappa_w = \frac{1}{2} \sum_{(x,y)\in E} w(x,y)(\frac{1}{\|\nabla u(x)\|} + \frac{1}{\|\nabla u(y)\|})(u(x) - u(y)).$$

Note that in the case of unweighted mesh graph, κ_w becomes a numerical discretization of mean curvature. This curvature, κ_w, is also used in [30] as a regularizer in a graph adaptation of the Chan-Vese method. In their work in progress [104], Van Gennip et al. propose a different definition of mean curvature on graphs and prove convergence of the MBO scheme on graphs.

Previous Work Using Graphs

Graph-based formulations have been used extensively for image processing applications [10, 26, 27, 36, 56, 57, 59, 74]. Interesting connections between these different algorithms, as well as between continuous and discrete optimizations, have been established in the literature. Grady has proposed a random walk algorithm [57] that performs interactive image segmentation using the solution to a combinatorial Dirichlet problem. Elmoataz et al. have developed generalizations of the graph Laplacian [36] for image denoising and manifold smoothing.

Couprie et al. in [26] define a conveniently parameterized graph-based energy function that is able to unify graph cuts, random walker, shortest paths and watershed optimizations. There, the authors test different seeded image segmentation algorithms, and discuss possibilities to optimize more general models with applications beyond image segmentation. In [27], alternative graph-based formulations of the continuous max-flow problem are compared, and it is shown that not all the properties satisfied in the continuous setting carry over to the discrete graph representation. For general data segmentation, Bresson et al. in [12], present rigorous convergence results for two algorithms that solve the relaxed Cheeger cut minimization, and show a formula that gives the correspondence between the global minimizers of the relaxed problem and the global minimizers of the combinatorial problem.

2.2 Graphical Framework, Extended

For the MBO method, we only need to define the Laplace operator in a more general graphical framework, since this is the only operator encountered in the procedure. For each of the versions of the method developed in Chapter 4, however, we need to consider more operators, and thus we outline the graphical framework in more detail here, giving more general definitions for other operators. We define operators on graphs in a similar fashion as done in [62, 103], where the justification for these choices is shown.

Assume m is the number of vertices in the graph and let $\mathcal{V} \cong \mathbb{R}^m$ and $\mathcal{E} \cong \mathbb{R}^{\frac{m(m-1)}{2}}$ be Hilbert spaces (associated with the set of vertices and edges, respectively) defined via the following inner products:

$$\langle u, \gamma \rangle_V = \sum_x u(x)\gamma(x)d(x)^r,$$

$$\langle \psi, \phi \rangle_{\mathcal{E}} = \frac{1}{2}\sum_{x,y} \psi(x,y)\phi(x,y)w(x,y)^{2q-1}$$

for some $r \in [0,1]$ and $q \in [\frac{1}{2}, 1]$. Let us also define the following norms:

$$\|u\|_{\mathcal{V}} = \sqrt{\langle u, u \rangle_{\mathcal{V}}} = \sqrt{\sum_x u(x)^2 d(x)^r},$$

$$\|\phi\|_{\mathcal{E}} = \sqrt{\langle \phi, \phi \rangle_{\mathcal{E}}} = \sqrt{\frac{1}{2}\sum_{x,y} \phi(x,y)^2 w(x,y)^{2q-1}},$$

$$\|\phi\|_{\mathcal{E},\infty} = \max_{x,y} |\phi(x,y)|.$$

The gradient operator $\nabla : \mathcal{V} \to \mathcal{E}$ is then defined as:

$$(\nabla u)_w(x,y) = w(x,y)^{1-q}(u(y) - u(x)). \tag{3}$$

The Dirichlet energy does not depend on r or q:

$$\frac{1}{2}\|\nabla u\|_{\mathcal{E}}^2 = \frac{1}{4}\sum_{x,y} w(x,y)(u(x) - u(y))^2.$$

The divergence $div : \mathcal{E} \to \mathcal{V}$ is defined as the adjoint of the gradient:

$$(\operatorname{div}_w \phi)(x) = \frac{1}{2d(x)^r} \sum_y w(x,y)^q (\phi(x,y) - \phi(y,x)), \tag{4}$$

where we define the adjoint using the following definition: $\langle \nabla u, \phi \rangle_{\mathcal{E}} = -\langle u, \operatorname{div}_w \phi \rangle_{\mathcal{V}}$.

We now have a family of graph Laplacians $\triangle_r = \operatorname{div}_w \dot{\nabla} : \mathcal{V} \to \mathcal{V}$:

$$(\triangle_w u)(x) = \sum_y \frac{w(x,y)}{d(x)^r}(u(y) - u(x)). \tag{5}$$

Viewing u as a vector in $\mathbb{R}^m$, we can write

$$-\triangle_w u = (\mathbf{D}^{1-r} - \mathbf{D}^{-r}\mathbf{W})u.$$

The case with $r = 0$ is the unnormalized Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{W}.$$

However, the matrix $\mathbf{L}$ is usually scaled to guarantee convergence to the continuum differential operator in the limit of large sample size [9]. Although several versions exist, we consider two popular versions of the symmetric Laplacian

$$\mathbf{L_s} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$$

and the random walk Laplacian ($r = 1$)

$$\mathbf{L_{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D^{-1}}\mathbf{W}.$$

The advantage of the former formulation is its symmetric property which allows for more efficient implementations.

A family of anisotropic total variations $TV_w : \mathcal{V} \to \mathbb{R}$ can now be defined:

$$TV_w(u) = \max\left\{\langle \operatorname{div}_w \phi, u \rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \leq 1\right\} = \frac{1}{2}\sum_{x.y} w(x,y)^q |u(x) - u(y)|. \tag{6}$$

Lastly, in this section, we consider the following graph-based Ginzburg Landau functional:

$$GL_\epsilon(u) = \|\nabla u\|_{\mathcal{E}}^2 + \frac{1}{\epsilon}\sum_x W(u(x)).$$

Remark. *It is noted in [103] that although the first term in the continuous Ginzburg-Landau functional*

$$\epsilon \int |\nabla u|^2 dx + \frac{1}{\epsilon}\int W(u) dx$$

is scaled by ϵ, the first term of GL_ϵ contains no ϵ. This occurs because the Dirichlet energy in the continuous Ginzburg-Landau functional is unbounded for functions of bounded variation and taking on two values of the minima of the double-well potential (almost everywhere). However, the difference terms of GL_ϵ are finite even in the case of binary functions, and no rescaling of the first term is necessary.

It remains to choose the parameters q and r. We choose $q = 1$ as in [103], where it is shown that for any r, TV_w is the Γ-limit (Gamma convergence) of a sequence of graph-based Ginzburg-Landau (GL)-type functionals:

Theorem 1. *$GL_\epsilon \xrightarrow{\Gamma} GL_0$ as $\epsilon \to 0$, where*

$$GL_\epsilon(u) = \|\nabla u\|_{\mathcal{E}}^2 + \frac{1}{\epsilon}\sum_x W(u(x)) = \frac{1}{2}\sum_{x,y} w(x,y)(u(x) - u(y))^2 + \frac{1}{\epsilon}\sum_x W(u(x)),$$

$$GL_0(u) = \begin{cases} TV_w(u)\textit{with q=1} & \textit{for } u \textit{ s.t. } u(x) \in \{0,1\} \\ \infty & \textit{otherwise} \end{cases}$$

Proof See Theorem 3.1 of [103]. ■

It is also shown in the paper (specifically Theorem 3.6) that *the addition of a fidelity term is compatible with Γ-convergence*. Since one of the algorithms we compare our methods to involves the Ginzburg-Landau functional, to be consistent, we use the above definitions with $q = 1$ in our formulations.

We choose $r = 1$ because it results in a normalized random walk Laplacian and the eigenvectors as well as the corresponding eigenvalues of the matrix can be efficiently calculated. Although the random walk Laplacian matrix itself is not symmetric, spectral graph theory described in [24] shows that the eigenvectors of the random walk Laplacian can be directly computed from knowing the diagonal matrix $\mathbf{D}$ and the eigenvectors of the symmetric graph Laplacian (which is a symmetric matrix) $\mathbf{L_s}$. In particular, λ is an eigenvalue of $\mathbf{L_{rw}}$ with eigenvector u if and only if λ is an eigenvalue of $\mathbf{L_s}$ with eigenvector $w = \mathbf{D}^{\frac{1}{2}}u$. This is proved by multiplying the eigenvalue equation $\mathbf{L_{rw}}u = \lambda u$ by $\mathbf{D}^{\mathbf{\frac{1}{2}}}$ from the left and then substituting $w = \mathbf{D}^{\frac{1}{2}}u$, obtaining $\mathbf{L_s}w = \lambda w$.

We take advantage of this property by calculating the eigenvalues and eigenvectors of the symmetric graph Laplacian (since symmetric matrices allow for more efficient implementations) and then using this information to calculate the same for the random walk Laplacian.

To summarize, we use the above operator definitions with $q = 1$ and $r = 1$.

Therefore, we use

$$TV_w(u) = \max\left\{\langle \operatorname{div}_w \phi, u\rangle_{\mathcal{V}} : \phi \in \mathcal{E}, \|\phi\|_{\mathcal{E},\infty} \leq 1\right\} = \frac{1}{2}\sum_{x.y} w(x,y)|u(x) - u(y)|. \qquad (7)$$

In this section, we use the notation $u(x)$ to denote the value of u at node $x \in V$ that provides information about the class membership of the node. Specifically, we use $u(x) = 0$ to denote the fact that node x belongs to class 1, and $u(x) = 1$ to denote that it belongs to class 2.

2.3 Optimization

Constrained and Unconstrained Minima

This section will focus on the connection between constrained and unconstrained minima. Consider the problem of

$$\min_x f(x)$$

under the constraint

$$g(x) = 0.$$

Suppose that this minimum is achieved at x_0. By the theory of Lagrange multipliers, there exists a multiplier λ such that, if $G = f + \lambda g$,

$$G'(x_0) = 0, \quad g(x_0) = 0,$$

$$G''(x_0, h) = \sum_{i,j=1}^{m} (\frac{\partial^2 G}{\partial x^i \partial x^j}) h^i h^j \geq 0$$

for all $h \neq 0$ such that

$$g'(x_0, h) = g'(x_0) \cdot h = 0.$$

A point y is considered non-singular if

$$\begin{vmatrix} \frac{\partial^2 G}{\partial x^i \partial x^j} & \frac{\partial g}{\partial x^i} \\ \frac{\partial g}{\partial x^j} & 0 \end{vmatrix}$$

is $\neq 0$ at y.

If x_0 is a nonsingular minimum point for f subject to $g = 0$, we see that $G''(x_0, h) = 0$ if and only if $h = 0$, and otherwise it is positive. Therefore, there exists a positive number c such that

$$G''(x_0, h) + c(g'(x_0, h))^2 > 0$$

for all $h \neq 0$. Let

$$F = f + \lambda g + \frac{1}{2}cg^2 = G + \frac{1}{2}cg^2.$$

Then we have the following equations:

$$F'(x_0) = G'(x_0) = 0,$$

$$F''(x_0, h) = G''(x_0, h) + c(g'(x_0, h))^2 > 0, h \neq 0.$$

Therefore, we have the following observation:

Note: If x_0 is a nonsingluar minimum of f subject to the constraint $g = 0$, there exists a multiplier λ and a constant c such that x_0 is an unconstrained local minimum of the function

$$F = f + \lambda g + \frac{1}{2}cg^2.$$

We also have a converse statement: if $g(x_0) = 0$ and x_0 is a minimum of such F, then x_0 is a minimum of f subject to the constraint $g = 0$.

This idea is used in the augmented Lagrangian method, to be discussed shortly.

Penalty Methods

A popular method to find a constrained minimum of a function is the method of penalty functions. Suppose one seeks the minimum of function f

$$\min_x f(x)$$

under the constraint $g(x) = 0$. The penalty method is an iterative procedure that, at iteration n, seeks the minimum point x_n of the function

$$F_n(x) = f(x) + \frac{1}{2}ng^2(x).$$

The limit of the sequence x_n, if it exists, is the constrained minimum of function f. Note also that

$$0 = F_n'(x_n) = f'(x_n) + ng(x_n)g'(x_n).$$

Considering the above, $\lambda^n = ng(x_n)$ converges to a Lagrange multiplier λ (see previous section) for the constrained minimum of f. However, this method has a disadvantage as it becomes sensitive to round-off errors due to the $ng^2(x)$ term since it is difficult to obtain an accurate representation of x_n for large values of n. To circumvent this difficulty, we present the augmented Lagrangian method.

Augmented Lagrangian Method

Consider the problem of finding the minimum point of f subject to the constraint $g(x) = 0$. One version of the augmented Lagrangian method is an iterative procedue that, at interation n, seeks the minimum point x_n of the function

$$F_n(x) = f(x) + \lambda^n g(x) + \frac{1}{2}cg^2(x).$$

The limit of the sequence x_n, if it exists, is the constrained minimum of the function f. To derive the update equation for λ, consider the following equation:

$$F'(x_n) = f'(x_n) + (\lambda^n + cg(x_n))g'(x_n) = 0.$$

So, one can consider selecting

$$\lambda^{n+1} = \lambda^n + cg(x_n).$$

In our case, we keep c constant. However, that does not have to be the case, and one can certainly derive a procedure for which c is updated at each step.

Translating this problem to the multi-constraint case, consider the problem of finding the minimum of f subject to constraints $g_i(x) = 0$ for $i = 1 : N$. Similar theory can be formulated using

$$F(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x) + \frac{1}{2}c\sum_i g_i^2(x)$$

with the λ^i being calculated in the following manner:

$$\lambda_i^{n+1} = \lambda_i^n + cg_i(x_n).$$

Therefore, we have the following augmented Lagrangian method:

Augmented Lagrangian Method: Consider the problem of finding the minimum of f subject to constraints $g_i(x) = 0$ for $i = 1 : N$. Let

$$F(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x) + \frac{1}{2}c\sum_i g_i^2(x).$$

Starting with an initial guess λ_i^1 for $i = 1 : N$, having obtained λ_i^n, let x^n be the minimum of $F_n = F(x, \lambda^n)$. We update λ by

$$\lambda_i^{n+1} = \lambda_i^n + cg_i(x_n).$$

As n increases, x^n converges the actual minima of f subject to the constraints.

If the problem is to find the maximum of f instead, we update λ by the following:

$$\lambda_i^{n+1} = \lambda_i^n - cg_i(x_n).$$

This can be easily derived from the previously stated theory, since we now use the function

$$F(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x) - \frac{1}{2}c\sum_i g_i^2(x).$$

The derivation is very similar to the one already shown. Suppose that this maximum is achieved at x_0. By the theory of Lagrange multipliers, there exists a multiplier λ such that, if $G = f + \lambda g$,

$$G'(x_0) = 0, \quad g(x_0) = 0,$$

$$G''(x_0, h) = \sum_{i,j=1}^{m} (\frac{\partial^2 G}{\partial x^i \partial x^j}) h^i h^j \leq 0$$

for all $h \neq 0$ such that

$$g'(x_0, h) = g'(x_0) \cdot h = 0.$$

There also exists a positive number c such that

$$G''(x_0, h) + c(g'(x_0, h))^2 < 0.$$

This justifies the use of F.

The advantage of this method is that unlike the penalty method, the function F_n doesn't contain a large constant in front of the g^2 term, in which case it would be difficult to obtain a good numerical approximation of the minimum for large n. Instead, the two terms allow the multipliers to stay a relatively small size. However, a disadvantage of the augmented Lagrangian method is that one needs to have an initial estimate of the multipliers, which is not the case with the penalty methods.

Convex Optimization and Lagrange Duality

Consider the following problem:

$$\min_x f(x)$$

subject to

$$h_i(x) \leq 0, \quad g_i(x) = 0 \tag{8}$$

for $i = 1 : N$. This is the *primal problem*. Let p^* denote the optimal value given by x^*:

$$p^* = \min_x f(x).$$

Now let

$$L(x,\lambda,\mu) = f(x) + \sum_i \lambda_i h_i(x) + \sum_i \mu_i g_i(x), \lambda_i \geq 0$$

and

$$k(\lambda,\mu) = \min_x L(x,\lambda,\mu), \qquad \lambda \geq 0, \mu.$$

We have the following inequality:

$$k(\lambda,\mu) \leq p^*.$$

This is because $k(\lambda,\mu) \leq L(x,\lambda,\mu) \leq f(x)$ for all x such that satisfy the constraints (8).

Now we introduce the *dual problem*

$$d^* = \max_{\lambda \geq 0, \mu} k(\lambda,\mu),$$

so d^* is the optimal value.

Since $k(\lambda,\mu) \leq p^*$, we have

$$d^* \leq p^*.$$

This is called the notion of $weakduality$ and it always holds. The notion of $strongduality$ occurs when $d^* = p^*$. Strong duality does not always occur, but it is true in the case when convexity is mixed with certain conditions, such as Slater condition or the KKT condition.

Now suppose we don't include the equality constraints. We can write the primal problem as

$$p^* = \min_x \max_{\lambda \geq 0} L(x,\lambda),$$

since maximizing $L(x,\lambda)$ in the variable λ gives the original function in this case. By definition, the dual problem is

$$d^* = \max_{\lambda \geq 0} \min_x L(x,\lambda).$$

So the principle of weak duality states that

$$\max_{\lambda \geq 0} \min_x L(x,\lambda) \leq \min_x \max_{\lambda \geq 0} L(x,\lambda).$$

We thus have some form of the max-min inequality. The principle of strong duality states that

$$\max_{\lambda \geq 0} \min_x L(x,\lambda) = \min_x \max_{\lambda \geq 0} L(x,\lambda).$$

In further sections, we include algorithms that deal directly with the primal problem, but also some that solve the dual problem.

2.4 Clustering

Considering the problem of finding the minimum cut (1) as way to cluster a set into two parts. As mentioned earlier, in order to avoid an undesired classification of an isolated vertex, one often needs to use some normalization. In particular, we mention the ratio cut and the normalized cut.

Ratio Cut

One way to modify the problem is to find a subset S of V such that

$$RatioCut(S,\bar{S}) = cut(S,\bar{S})\Big(\frac{1}{|S|}+\frac{1}{|\bar{S}|}\Big)$$

is minimized. This is a NP hard discrete problem [105]. One way to simplify it would be to allow the solution to take arbitrary values in $\mathbb{R}$. This leads to the following relaxed RatioCut problem:

$$\min_{u\in R^n}\langle u,\mathbf{L}u\rangle, \quad u\perp\mathbf{1}, \quad ||u||^2=n.$$

The fact that the above problem obtains a real-valued solution instead of a discrete-valued solution is emphasized. To solve the above problem, one can apply the Raleigh-Ritz theorem, and the solution is given by the second eigenvector of the Laplacian. To obtain a binary solution, one can use several methods, the simplest of which is thresholding.

Normalized Cut

If we let $vol(S)=\sum_{x\in S}d(x)$, where $d(x)$ represents the degree of vertex x, another way to modify the problem is to find a subset S of V such that

$$Ncut(S,\bar{S}) = cut(S,\bar{S})\Big(\frac{1}{vol(S)}+\frac{1}{vol(\bar{S})}\Big)$$

is minimized. This is yet again a NP hard discrete problem [105]. We simplify it by allowing the solution to take arbitrary values in $\mathbb{R}$. This leads to the following relaxed Ncut problem:

$$\min_{u\in R^n}\langle u,\mathbf{L}u\rangle, \quad Du\perp\mathbf{1}, \quad \langle u,\mathbf{D}u\rangle = vol(Y).$$

To solve the above problem, one can apply the Raleigh-Ritz theorem, and the solution is given by the second eigenvector of the random walk Laplacian. Thresholding can be used to obtain a binary solution.

Other Normalizations

Two other possible normalizations are

$$N1 = \frac{cut(S, \bar{S})}{min(vol(S), vol(\bar{S}))},$$

$$N2 = \frac{cut(S, \bar{S})}{min(|S|, |\bar{S}|)}.$$

Spectral Clustering

We have seen that the ratio cut and the normalized cut problems can be formulated in a continuous setting, with solutions given by the second eigenvectors of the Laplacian ($\mathbf{L} = \mathbf{D} - \mathbf{W}$) and the random walk Laplacian ($\mathbf{L_{rw}} = \mathbf{D^{-1}L}$), respectively. For binary problems, one can simply find the solution by thresholding the second eigenvector of an appropriate Laplacian. Now, suppose there are k clusters. The method of spectral clustering computes the k clusters in the following manner:

1. Formulate the data set in a graph setting.
2. Compute either the Laplacian ($\mathbf{L} = \mathbf{D} - \mathbf{W}$) and the random walk Laplacian ($\mathbf{L_{rw}} = \mathbf{D^{-1}L}$)
3. Compute the first k eigenvectors (v_1,v_2,..., v_k) of the Laplacian (or the random walk Laplacian)
4. Let $\mathbf{U}$ be the matrix containing the vectors v_1, v_2,..., v_k as columns.
5. Cluster the rows of the matrix $\mathbf{U}$ with the k-means algorithm into k clusters.

The k-means algorithm for k clusters proceeds iteratively by first choosing k means and then assigning each point to the cluster whose mean is the “closest” to the point. The mean of each cluster is then recalculated. The iterations continue until there is little change from one iteration to the next.

CHAPTER 3

MBO Method

3.1 Derivation of the Method

We consider the general problem for data classification and image processing:

$$\min_u \{E(u) = R(u) + F(u)\},$$

where $R(u)$ is the regularization functional and $F(u)$ is the fidelity term. Here u is a function defined on the space of the data set and describes the appropriate characteristic. For example, in the case of the problem of classification, u indicates the class number. In the case of image inpainting, u indicates intensity value of a pixel. Some common $R(u)$ regularization term examples are

$$R(u) = \int |\nabla u| dx, \qquad R(u) = \int |\nabla u|^2 dx.$$

Some common $F(u)$ fidelity term examples are

$$F(u) = \int \lambda(x)|u - u_0| dx, \qquad F(u) = \int \lambda(x)|u - u_0|^2 dx.$$

The λ term regulates the fidelity region. It equal to 0 on the non-fidelity region and to a constant on the fidelity region. For example, in the case of image inpainting, the fidelity region is everything but the damaged region. In the case of classification, the fidelity would be all the points with known classification values. The resulting functional is a tradeoff between accuracy in the classification of given labels and function smoothness. It is also desirable to choose R to preserve the sharp discontinuities that may arise in the boundaries between classes.

To formulate the MBO method for classification and image processing, we introduce another example of the regularization term $R(u) =$ Ginzburg-Landau functional:

$$GL_\epsilon(u) = \frac{\epsilon}{2} \int |\nabla u|^2 dx + \frac{1}{\epsilon} \int W(u) dx, \tag{9}$$

where $W(u) = (u^2 - 1)^2$. We use this regularization term for the MBO method. We also use the L^2 fidelity term.

Ginzburg-Landau Functional

The classical Ginzburg-Landau (GL) functional was originally proposed to describe physical phenomena such as liquid-gas transitions and superconductivity [97]. Here, u is a scalar field defined over a space of arbitrary dimensionality and representing the state of the phases in the system, ∇ denotes the spatial gradient operator, $W(u)$ is a double-well potential, such as $W(u) = \frac{1}{4}(u^2-1)^2$ and ϵ is a positive constant.

The functional (9) is composed of two terms: a smoothing term that measures the differences in the components of the field, and a potential term that measures how far each component is from a specific value (± 1 in the example above). Consequently, the minimization of the first term leads to smoother regions, while the minimization of the second penalizes variations from the minima of the double-well potential. Given initial conditions with states $+1$ and states -1 distributed randomly in the domain, the minimization of the GL functional entails an inherent conflict between the two terms in the functional, leading to the generation of a transition region: the diffuse interface. The smoothness of this diffuse interface is regulated by the parameter ϵ. For small ϵ, the state minimizing the functional contains sharp transitions between the minima of the double-well potential, while a large ϵ gives more weight to the smoothing term so that the transitions are more gradual.

It is shown in [70] that the $\epsilon \to 0$ limit of the GL functional, in the sense of Γ-convergence, is the Total Variation (TV) semi-norm, so one can write:

$$GL_\epsilon(u) \to_\Gamma ||u||_{TV},$$

where

$$||u||_{TV} = \int_\Omega |\nabla u| dx.$$

Therefore, the justification for using the justification for using the Ginzburg-Landau functional comes from the fact that it is related to the total variation seminorm, which has been used successfully in many image processing applications. It has also been applied to numerical analysis of differential equations [60]. The graph version of this result was shown in [103].

This convergence allows the two functionals to be interchanged in some cases. One might prefer to use the GL functional instead of the TV semi-norm since its highest order term is purely quadratic which allows for efficient minimization procedures. It also allows us to use spectral mehtods for minimization. In contrast, minimization of the TV semi-norm leads to a nonlinear curvature term, making it less trivial to solve numerically. However, recent advances, such as the split Bregman method described in [55], have made progress in such problems.

An analogous convergence property has recently been shown in the case of graphs as well, for binary segmentations [103]. Since TV is an L^1-based metric, TV-minimization leads to sparse solutions, namely indicator functions that closely resemble the discrete solution of the original NP-hard combinatorial segmentation problem [100]. Thus, the GL functional actually becomes an L^1 metric in the small ϵ limit, and leads to sharp transitions between classes. Intuitively, the convergence of GL to TV holds because in the limit of a vanishing interface, the potential takes precedence and the graph nodes are forced towards the minima of the potential, achieving a configuration of minimal length of transition. This is contrast to more traditional spectral clustering approaches, which can be understood as L^2-based methods and do not favor sparse solutions. Furthermore, while the smoothness of the transition in the GL functional is regulated by ϵ, in practice the value of ϵ does not have to be decreased all the way to zero to obtain sharp transitions. This capability of modeling the separation of a domain into regions or phases with a controlled smoothness transition between them makes the diffuse interface description attractive for segmentation problems, and distinguishes it from more traditional graph-based spectral partitioning methods.

The diffuse interface description, with its controlled smoothness transition between phases, is attractive for segmentation problems due to its straightforward way of modeling the separation of a domain into regions or phases. It has been used successfully in image impainting [8,32] and image segmentation [39].

Derivation

By using the Ginzburg-Landau functional for the regularization term and the L^2 fidelity term, we obtain the following minimization problem:

$$\min_u \big\{E(u) = \frac{\epsilon}{2}\int |\nabla u|^2 dx + \frac{1}{\epsilon}\int W(u)dx + \int \lambda(x)|u-u_0|^2 dx\big\}. \tag{10}$$

The energy can be minimized in the L_2 sense using gradient descent. This leads to the following dynamic equation (*modified Allen-Cahn equation*):

$$\frac{\partial u}{\partial t} = -\frac{\delta GL_\epsilon}{\delta u} - \mu\frac{\delta F}{\delta u} = \epsilon\Delta u - \frac{1}{\epsilon}W'(u) - \mu\frac{\delta F}{\delta u}, \tag{11}$$

where Δ represents the Laplacian operator. A local minimizer of the energy is obtained by evolving this expression to steady state. Note that E is not convex, and may have multiple local minima.

In their work [9], Bertozzi and Flenner propose a segmentation algorithm for solving (5.1) in a graph setting. The functional is minimized using the method of gradient descent and convex splitting. The main purpose of thie MBO method is to develop a more efficient and simple method for minimizing (5.1) in the small ϵ limit. An answer comes from the relation between the Allen-Cahn equation and the motion by mean curvature.

Let us start by reviewing this connection in the continuous setting. In [81], Merriman, Bence and Osher propose an algorithm to approximate motion by mean curvature, or motion in which normal velocity equals mean curvature, using threshold dynamics. The authors note that if one applies the heat equation to an interface, then the diffusion blunts the sharp points of the boundary, but has very little effect on the flatter regions. Therefore, one can imagine that diffusion creates some sort of motion by mean curvature, providing that we specify the boundaries of the moving set.

Given a phase field $u(x, t)$, consider the basic (unmodified) Allen-Cahn equation, namely equation (11) without the fidelity term:

$$\frac{\partial u}{\partial t} = \epsilon \Delta u - \frac{1}{\epsilon} W'(u). \tag{12}$$

For small values of ϵ, the following time-splitting scheme can be used numerically to evolve the Allen-Cahn equation:

1. The first step is propagation using:
$$\frac{\partial u}{\partial t} = \epsilon \Delta u.$$
2. The second step is propagation using:
$$\frac{\partial u}{\partial t} = -\frac{1}{\epsilon} W'(u).$$

Note, however, that in the $\epsilon \to 0$ limit, the second step is simply thresholding [81]. Thus, as $\epsilon \to 0$, the time splitting scheme above consists of alternating between diffusion and thresholding steps.

It has been shown [92] that in the limit $\epsilon \to 0$, the rescaled solutions $u_\epsilon(z, t/\epsilon)$ of (12) yield motion by mean curvature of the interface between the two phases of the solutions. This motivates the two sequential steps of the MBO scheme:

1. *Diffusion.* Let $u^{n+\frac{1}{2}} = S(\delta t) u^n$ where $S(\delta t)$ is the propagator (by time δt) of the standard heat equation:
$$\frac{\partial u}{\partial t} = \Delta u.$$

2. *Thresholding.* Let

$$u^{n+1} = \begin{cases} 1 & \text{if } u^{n+\frac{1}{2}} \geq 0, \\ -1 & \text{if } u^{n+\frac{1}{2}} < 0. \end{cases}$$

Barles [5] and Evans [40] have proven rigorously that this scheme approximates motion by mean curvature.

Multiple extensions, adaptations and applications of the MBO scheme are present in literature. We find the modification of the MBO scheme for solving the inhomogeneous Allen-Cahn equation proposed in [39] particularly interesting. To create a fast image segmentation algorithm, Esedoglu and Tsai propose a thresholding scheme for minimizing a diffuse interface version of the piecewise constant Mumford-Shah functional

$$MS_\epsilon(u, c_1, c_2) = \int_D \epsilon|\nabla u|^2 + \frac{1}{\epsilon}W(u) + \lambda\{u^2(c_1 - f)^2 + (1-u)^2(c_2 - f)^2\}dx, \tag{13}$$

where f is the image. The first variation of the model (13) yields the following gradient descent equation:

$$u_t = 2\epsilon\Delta u - \frac{1}{\epsilon}W'(u) + 2\lambda\{u(c_1 - f)^2 + (1-u)(c_2 - f)^2\}$$

and the adaptation of the MBO scheme is used to solve it. Esedoglu and Tsai propose the following scheme (similar to the MBO scheme where the propagation step based on the heat equation is combined with thresholding):

* **Step 1** Let $v(x) = S(\delta t)u_n(x)$ where $S(\delta t)$ is a propagator by time δt of the equation:

$$w_t = \Delta w - 2\tilde{\lambda}\left(w(c_1 - f)^2 + (1-w)(c_2 - f)^2\right)$$

with appropriate boundary conditions.

* **Step 2** Set

$$u_{n+1}(x) = \begin{cases} 0 & \text{if } v(x) \in (-\infty, \frac{1}{2}], \\ 1 & \text{if } v(x) \in (\frac{1}{2}, \infty). \end{cases}$$

Some other extensions of the MBO scheme appeared in [37, 38, 83]. An efficient algorithm for motion by mean curvature using adaptive grids was proposed in [93].

The motion by mean curvature of the MBO scheme can be generalized to the case of functions on a graph in much the same way as the procedure followed for the modified Allen-Cahn equation (11). We now use the same ideas and apply a two-step time splitting scheme to (11) so that the second step is the same as the one in the original MBO scheme. The idea is then to replace all the operators with a more general graph term, since we are considering the graphical framework. The only operator to deal with here, is the Δ operator, and we can replace it by several different versions of the graph Laplacian. In the graphical framework, we have the following three versions that are related to the differential Δ operator:

* $\mathbf{L} = \mathbf{D} - \mathbf{W}$, unnormalized Laplacian

* $\mathbf{L_s} = \mathbf{D^{-\frac{1}{2}} L D^{-\frac{1}{2}}}$, symmetric Laplacian

* $\mathbf{L_{rw}} = \mathbf{D^{-1} L}$, random walk Laplacian

Since $\mathbf{L_s}$ is a symmetric matrix, we use the symmetric Laplacian, and thus replace Δu by $-\mathbf{L_s}\boldsymbol{u}$. This connection is to be explained in Section 10.

The discretized version of the algorithm is:

Binary MBO Algorithm:

Initialize u. Until convergence, alternate between the following two steps:

1. Heat equation with forcing term:

$$\frac{\boldsymbol{u}^{n+\frac{1}{2}} - \boldsymbol{u}^n}{dt} = -\mathbf{L_s}\boldsymbol{u}^n - \boldsymbol{\mu}(\boldsymbol{u}^n - \hat{\boldsymbol{u}}). \tag{14}$$

2. Thresholding:

$$u_i^{n+1} = \begin{cases} 1, & \text{if } u_i^{n+\frac{1}{2}} > 0, \\ -1, & \text{if } u_i^{n+\frac{1}{2}} < 0. \end{cases}$$

Here, after the second step, u_i^n can take only two values of 1 or -1; thus, this method is appropriate for binary segmentation. Note that the fidelity term scaling can be different from the one in (11).

The first part of the two step scheme is solved using the spectral decomposition of the symmetric graph Laplacian. Let $u^n = \sum_k a_k^n \phi_k(x)$ and $C_1\lambda(u^n - u_0) = \sum_k d_k^n \phi_k(x)$, where $\phi(x)$ are the eigenfunctions of the symmetric Laplacian. Using the obtained representations and equation (30), we obtain

$$a_k^{n+1} = \frac{a_k^n - dt d_k^n}{1 + dt\lambda_k},$$

where λ_k are the eigenvalues of the symmetric graph Laplacian.

In practice, it can be productive to repeat the diffusion step a number of times before thresholding. In order to keep the convention that one iteration of the diffusion-thresholding procedure corresponds to one time step, we divide dt by the number of diffusion steps per iteration, which we denote N_S.

To compute the eigenvalues and eigenvectors of the graph Laplacian, we use two methods. One of them is the Raleigh-Chebyshev procedure of [1] and the second one is the Nyström extension [9, 44, 45]. See the Appendix for a brief description of the latter method.

3.2 MBO Method Procedure

Here are the steps of the MBO method in finding a binary minimizer u.

* Create a graph from the data, choose a weight function and then calculate the symmetric graph Laplacian.

* Calculate the eigenvectors and eigenvalues of the symmetric graph Laplacian.

* Initialize u.

* Find or set the fidelity region.

* Apply the two-step scheme described earlier until a stopping criterion is satisfied.

The final u will be binary. Changes for multiclass case will be discussed in the next section.

3.3 Extension to the Multiclass Case

Given N_D data points, we generalize the label vector $\boldsymbol{u}$ to a label matrix $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{N_D})^T$. Rather than node i adopting a single state $u_i \in \mathbb{R}$, it now adopts a composition of states expressed by a vector $\mathbf{u}_i \in \mathbb{R}^K$ where the kth component of $\mathbf{u}_i$ is the strength with which it takes on class k. The matrix $\mathbf{U}$ has dimensions $N_D \times K$, where K is the total number of possible classes.

For each node i, we require the vector $\mathbf{u}_i$ to be an element of the Gibbs simplex Σ^K, defined as

$$\Sigma^K := \left\{ (x_1, \ldots, x_K) \in [0,1]^K \,\middle|\, \sum_{k=1}^{K} x_k = 1 \right\}.$$

Vertex k of the simplex is given by the unit vector $\boldsymbol{e}_k$, whose kth component equals 1 and all other components vanish. These vertices correspond to pure phases, where the node belongs exclusively to class k. Note that the simplex formulation has a straightforward probabilistic interpretation, with $\mathbf{u}_i$ representing the probability distribution over the K classes. In other segmentation algorithms, such as spectral clustering, these real-valued variables can have different interpretations that are exploited for specific applications, as discussed in [59, 74].

3.3.1 Version 1: The MBO Method Extension (Multiclass MBO)

Using the standard Gibbs-simplex Σ^K just defined, the multiclass extension of the algorithm in [80] is straightforward. The notation is the same as in the previous section: we use a matrix $\mathbf{U}$ to represent the phase composition of nodes. The second step of the algorithm is modified, however, so that the thresholding is converted to the displacement of the vector field variable towards the closest vertex in the Gibbs simplex. In other words, now in the second step the row vector ${\mathbf{u}_i}^{n+\frac{1}{2}}$ of step 1 is projected back to the simplex (using the approach outlined in [23] as before) and then a pure phase given by the vertex in the Σ^K simplex closest to ${\mathbf{u}_i}^{n+\frac{1}{2}}$ is assigned to be the new phase composition of node i.

In summary, the new algorithm consists of alternating between the following two steps to obtain approximate solutions $\mathbf{U}^n$ at discrete times:

1. Heat equation with forcing term:
$$\frac{\mathbf{U}^{n+\frac{1}{2}} - \mathbf{U}^n}{dt} = -\mathbf{L}_s \mathbf{U}^n - \boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}}). \tag{15}$$

2. Thresholding:
$${\mathbf{u}_i}^{n+1} = \boldsymbol{e}_k,$$
where vertex $\boldsymbol{e}_k$ is the vertex in the simplex closest to $projectToSimplex({\mathbf{u}_i}^{n+\frac{1}{2}})$.

As for the multiclass GL algorithm, when a label is known, it is represented by the corresponding vertex in the Σ^K simplex. The final classification is achieved by assigning node i to class k if if the kth component of $\mathbf{u}_i$ is one. Again, as in the binary case, the diffusion step can be repeated a number of times before thresholding and when that happens, dt is divided by the number of diffusion iterations N_S.

As in the previous section, we use an implicit numerical scheme. For the MBO algorithm, the procedure involves modifying (19) to apply $\mathbf{L}_s$ to $\mathbf{U}^{n+\frac{1}{2}}$ instead of to $\mathbf{U}^n$. This gives the diffusion step
$$\mathbf{U}^{n+\frac{1}{2}} = \mathbf{B}^{-1}\left[\mathbf{U}^n - dt\, \boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}})\right],$$

where

$$\mathbf{B} = \mathbf{I} + dt\,\mathbf{L}_{\boldsymbol{s}}.$$

As before, we use the eigendecomposition $\mathbf{L}_{\boldsymbol{s}} = \mathbf{X}\boldsymbol{\Lambda}\mathbf{X}^T$ to write

$$\mathbf{B} = \mathbf{X}\left(\mathbf{I} + dt\,\boldsymbol{\Lambda}\right)\mathbf{X}^T,$$

which we approximate using the first N_e eigenfunctions.

For initialization, the phase compositions of the fidelity points are set to the vertices of the simplex corresponding to the known labels, while the phase compositions of the rest of the points are set randomly.

The energy minimization proceeds until a steady state condition is reached. Once the change of the norm of the vector field in subsequent iterations falls below a threshold, the system is no longer evolving and the energy decrement is negligible. Consequently, the calculation is stopped when

$$\frac{\max_i \|\mathbf{u}_i{}^{n+1} - \mathbf{u}_i{}^{n}\|^2}{\max_i \|\mathbf{u}_i{}^{n+1}\|^2} < \eta,$$

where η represents a given small positive constant. The final classes are obtained by assigning class k to node i if $\mathbf{u}_i$ is closest to vertex $\boldsymbol{e}_k$ on the Gibbs simplex.

The multiclass MBO algorithm is summarized in Figure 1. Its complexity is $O(N_D K N_e N_S)$ operations for the main loop, $O(N_D K \log K)$ operations for the projection to the simplex and $O(N_D K)$ operations for thresholding. As for the multiclass GL algorithm, $N_e \ll N_D$ and $K \ll N_D$. Furthermore N_S needs to be set to three, and due to the thresholding step, we find that extremely few iterations (e.g., 6) are needed to reach steady state. Thus, in practice, the complexity of this algorithm is linear as well, and typical runtimes are very rapid as shown in Table III.

Note that graph analogues of continuum operators, such as gradient and Laplacian, can be constructed using tools of nonlocal discrete calculus. Hence, it is possible to express notions of graph curvature for arbitrary graphs, even with no geometric embedding, but this is not straightforward. For a more detailed discussion about the MBO scheme and motion by mean curvature on graphs, we refer the reader to [104].

3.3.2 Version 2: A Ginzburg-Landau Multiclass Extension (Multiclass GL)

Here we provide a different version of the multiclass extension that does not use the MBO scheme.

Figure 1: Multiclass MBO Algorithm

```
**Require:** $dt, N_D, N_e, N_S, K, \boldsymbol{\mu}, \hat{\mathbf{U}}, \boldsymbol{\Lambda}, \mathbf{X}$
**Ensure:** $\text{out} = \mathbf{U}^{\mathbf{end}}$
  $\mathbf{Y} \leftarrow \left(\mathbf{I} + \frac{dt}{N_S}\boldsymbol{\Lambda}\right)^{-1} \mathbf{X}^T$
  **for** $i = 1 \rightarrow N_D$ **do**
    $U_{ik}^0 \leftarrow rand((0,1)),\ \mathbf{u}_i{}^0 \leftarrow projectToSimplex(\mathbf{u}_i{}^0).$  If $\mu_i > 0,\ U_{ik}^0 \leftarrow \hat{U}_{ik}^0$
  **end for**
  $n \leftarrow 1$
  **while** Stop criterion not satisfied **do**
    **for** $s = 1 \rightarrow N_S$ **do**
      $\mathbf{Z} \leftarrow \mathbf{Y}\left[\mathbf{U}^n - \frac{dt}{N_S}\,\boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}})\right]$
      $\mathbf{U}^{n+1} \leftarrow \mathbf{X}\mathbf{Z}$
    **end for**
    **for** $i = 1 \rightarrow N_D$ **do**
      $\mathbf{u}_i{}^{n+1} \leftarrow projectToSimplex(\mathbf{u}_i{}^{n+1})$
      $\mathbf{u}_i{}^{n+1} \leftarrow \boldsymbol{e}_k$, where $k$ is closest simplex vertex to $\mathbf{u}_i{}^{n+1}$
    **end for**
    $n \leftarrow n + 1$
  **end while**
```

The multiclass GL energy functional for the phase field approach on graphs is written as:

$$\begin{aligned} E(\mathbf{U}) &= \frac{\epsilon}{2}\langle \mathbf{U}, \mathbf{L}_{\boldsymbol{s}}\mathbf{U}\rangle + \frac{1}{2\epsilon}\sum_{i\in V}\left(\prod_{k=1}^{K}\frac{1}{4}\left\|\mathbf{u}_i - \boldsymbol{e}_k\right\|_{L_1}^2\right) \\ &\quad + \sum_{i\in V}\frac{\mu_i}{2}\ \left\|\mathbf{u}_i - \hat{\mathbf{u}}_i\right\|^2, \end{aligned} \tag{16}$$

where

$$\langle \mathbf{U}, \mathbf{L}_{\boldsymbol{s}}\mathbf{U}\rangle = \text{trace}(\mathbf{U}^T\mathbf{L}_{\boldsymbol{s}}\mathbf{U}),$$

and $\hat{\mathbf{u}}_i$ is a vector indicating prior class knowledge of sample i. We set $\hat{\boldsymbol{u}}_i = \boldsymbol{e}_k$ if node i is known to be in class k.

As mentioned before, the first (smoothing) term in the GL functional (16) measures variations in the vector field. The simplex representation has the advantage that, like in Potts-based models but unlike in some other multiclass methods, the penalty assigned to differently labeled neighbors is independent of the integer ordering of the labels. The second (potential) term drives the system closer to the vertices of the simplex, with the use of an L_1 norm preventing the emergence of an undesirable minimum at the center of the simplex, as would happen with an L_2 norm for large K. This potential aims to provide a clear way to calculate class memberships, as the phase composition is purer near the vertices of the simplex. The compromise between the smoothing and potential terms is established through the constant ϵ. The third (fidelity) term enables the encoding of *a priori* information.

Note that one can obtain meaningful results without fidelity information (unsupervised), but the methods for doing so are not as straightforward. One example is a new TV-based modularity optimization method [66] that makes no assumption as to the number of classes and can be recast as GL minimization. Also, while Γ-convergence to TV in the graph setting has been proven for the binary segmentation problem [103], no similar convergence property has yet been proven for the multiclass case. We leave this as an open conjecture.

Following [9], we use a convex splitting scheme to minimize the GL functional in the phase field approach. The energy functional (16) is decomposed into convex and concave parts:

$$\begin{aligned}
E(\mathbf{U}) &= E_{\text{convex}}(\mathbf{U}) + E_{\text{concave}}(\mathbf{U}),\\
E_{\text{convex}}(\mathbf{U}) &= \frac{\epsilon}{2}\langle \mathbf{U}, \mathbf{L}_s \mathbf{U}\rangle + \frac{C}{2}\langle \mathbf{U}, \mathbf{U}\rangle,\\
E_{\text{concave}}(\mathbf{U}) &= \frac{1}{2\epsilon}\sum_{i\in V}\prod_{k=1}^{K}\frac{1}{4}\left\|\mathbf{u}_i - \boldsymbol{e}_k\right\|_{L_1}^2\\
&\quad + \sum_{i\in V}\frac{\mu_i}{2}\left\|\mathbf{u}_i - \hat{\mathbf{u}}_i\right\|_{L_2}^2 - \frac{C}{2}\langle \mathbf{U}, \mathbf{U}\rangle
\end{aligned}$$

with $C \in \mathbb{R}$ denoting a constant that is chosen to guarantee the convexity/concavity of the energy terms. Evaluating the second derivative of the partitions, and simplifying terms, yields:

$$C \geq \mu + \frac{1}{\epsilon}. \tag{17}$$

The convex splitting scheme results in an unconditionally stable time-discretization scheme using a gradient descent implicit in the convex partition and explicit in the concave partition, as given by the form [39, 41, 110]

$$U_{ik}^{n+1} + dt\,\frac{\delta E_{\text{convex}}}{\delta U_{ik}}(U_{ik}^{n+1}) = U_{ik}^{n} - dt\,\frac{\delta E_{\text{concave}}}{\delta U_{ik}}(U_{ik}^{n}).$$

We write this equation in matrix form as

$$\begin{aligned}\mathbf{U}^{n+1} &+ dt\left(\epsilon \mathbf{L_s}\mathbf{U}^{n+1} + C\mathbf{U}^{n+1}\right) \\ &= \mathbf{U}^n - dt\left(\frac{1}{2\epsilon}\mathbf{T}^n + \boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}}) - C\mathbf{U}^n\right),\end{aligned}$$

where

$$T_{ik} = \sum_{l=1}^{K} \frac{1}{2}\left(1 - 2\delta_{kl}\right)\left\|\mathbf{u}_i - \boldsymbol{e}_l\right\|_{L_1} \prod_{\substack{m=1\\ m\neq l}}^{K} \frac{1}{4}\left\|\mathbf{u}_i - \boldsymbol{e}_m\right\|_{L_1}^2,$$

$\boldsymbol{\mu}$ is a diagonal matrix with elements μ_i, and $\hat{\mathbf{U}} = (\hat{\mathbf{u}}_1, \ldots, \hat{\mathbf{u}}_{N_D})^T$.

Solving (18) for $\mathbf{U}^{n+1}$ gives the iteration equation

$$\mathbf{U}^{n+1} = \mathbf{B}^{-1}\left[\left(1 + C\ dt\right)\mathbf{U}^n - \frac{dt}{2\epsilon}\mathbf{T}^n - dt\,\boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}})\right],$$

where

$$\mathbf{B} = (1 + C\,dt)\mathbf{I} + \epsilon\,dt\,\mathbf{L_s}.$$

This implicit scheme allows the evolution of $\mathbf{U}$ to be numerically stable regardless of the time step dt, in spite of the numerical "stiffness" of the underlying differential equations which could otherwise force dt to be impractically small.

In general, after the update, the phase field is no longer on the Σ^K simplex. Consequently, we use the procedure in [23] to project back to the simplex.

Computationally, the scheme's numerical efficiency is increased by using a low-dimensional subspace spanned by only a small number of eigenfunctions. Let $\mathbf{X}$ be the matrix of eigenvectors of $\mathbf{L_s}$ and $\boldsymbol{\Lambda}$ be the diagonal matrix of corresponding eigenvalues. We now write $\mathbf{L_s}$ as its eigendecomposition $\mathbf{L_s} = \mathbf{X}\boldsymbol{\Lambda}\mathbf{X}^T$, and set

$$\mathbf{B} = \mathbf{X}\left[(1 + C\,dt)\mathbf{I} + \epsilon\,dt\,\boldsymbol{\Lambda}\right]\mathbf{X}^T,$$

but we approximate $\mathbf{X}$ by a truncated matrix retaining only N_e eigenvectors ($N_e \ll N_D$), to form a matrix of dimension $N_D \times N_e$. The term in brackets is simply a diagonal $N_e \times N_e$ matrix. This allows $\mathbf{B}$ to be calculated rapidly, but more importantly it allows the update step (3.3.2) to be decomposed into two significantly faster matrix multiplications (as discussed below), while sacrificing little accuracy in practice.

For initialization, the phase compositions of the fidelity points are set to the vertices of the simplex corresponding to the known labels, while the phase compositions of the rest of the points are set randomly.

Figure 2: Multiclass GL Algorithm

```
**Require:** $\epsilon, dt, N_D, N_e, K, \boldsymbol{\mu}, \hat{\mathbf{U}}, \boldsymbol{\Lambda}, \mathbf{X}$
**Ensure:** $\text{out} = \mathbf{U}^{\mathbf{end}}$
  $C \leftarrow \mu + \frac{1}{\epsilon}$
  $\mathbf{Y} \leftarrow \left[(1 + C\,dt)\mathbf{I} + \epsilon\,dt\boldsymbol{\Lambda}\right]^{-1}\mathbf{X}^T$
  for $i = 1 \rightarrow N_D$ do
    $U_{ik}^0 \leftarrow rand((0,1)),\ U_{ik}^0 \leftarrow projectToSimplex(\mathbf{u}_i{}^0)$.   If $\mu_i > 0,\ U_{ik}^0 \leftarrow \hat{U}_{ik}^0$
  end for
  $n \leftarrow 1$
  while Stop criterion not satisfied do
    for $i = 1 \rightarrow N_D,\ k = 1 \rightarrow K$ do
      $T_{ik}^n \leftarrow \sum_{l=1}^{K} \frac{1}{2}\left(1 - 2\delta_{kl}\right) \|\mathbf{u}_i{}^n - \boldsymbol{e}_l\|_{L_1} \prod_{m=1, m \neq l}^{K} \frac{1}{4} \|\mathbf{u}_i{}^n - \boldsymbol{e}_m\|_{L_1}^2$
    end for
    $\mathbf{Z} \leftarrow \mathbf{Y}\left[(1 + C\ dt)\,\mathbf{U}^n - \frac{dt}{2\epsilon}\,\mathbf{T}^n - dt\,\boldsymbol{\mu}(\mathbf{U}^n - \hat{\mathbf{U}})\right]$
    $\mathbf{U}^{n+1} \leftarrow \mathbf{XZ}$
    for $i = 1 \rightarrow N_D$ do
      $\mathbf{u}_i{}^{n+1} \leftarrow projectToSimplex(\mathbf{u}_i{}^{n+1})$
    end for
    $n \leftarrow n + 1$
  end while
```

The energy minimization proceeds until a steady state condition is reached. The final classes are obtained by assigning class k to node i if $\mathbf{u}_i$ is closest to vertex $\boldsymbol{e}_k$ on the Gibbs simplex. Consequently, the calculation is stopped when

$$\frac{\max_i \|\mathbf{u}_i{}^{n+1} - \mathbf{u}_i{}^n\|^2}{\max_i \|\mathbf{u}_i{}^{n+1}\|^2} < \eta,$$

where η represents a given small positive constant.

The algorithm is outlined in Figure 2. While other operator splitting methods have been studied for minimization problems (e.g. [73]), ours has the following advantages: (i) it is direct (i.e. it does not require the solution of further minimization problems), (ii) the resolution can be adjusted by increasing the number of eigenvectors N_e used in the representation of the phase field, and (iii) it has low complexity. To see this final point, observe that each iteration of the multiclass

GL algorithm has only $O(N_D K N_e)$ operations for the main loop, since matrix $\mathbf{Z}$ in Figure 2 only has dimensions $N_e \times K$, and then $O(N_D K \log K)$ operations for the projection to the simplex. Usually, $N_e \ll N_D$ and $K \ll N_D$, so the dominant factor is simply the size of the data set N_D. In addition, it is generally the case that the number of iterations required for convergence is moderate (around 50 iterations). Thus, practically speaking, the complexity of the algorithm is linear.

Note on Previous Work on Multiclass Classification

Not all the methods deal directly with the multiple classes in the data set. A different approach is to reduce the multiclass case to a series of two-class problems and to combine the sequence of resulting sub-classifications. Strategies employed include recursive partitioning, hierarchical classification and binary encodings, among others. For example, Dietterich and Bakiri use a binary approach to encode the class labels [31]. In [61], a pairwise coupling is described, in which each two-class problem is solved and then a class decision is made combining the decisions of all the subproblems. Szlam and Bresson present a method involving Cheeger cuts and split Bregman iteration [55] to build a recursive partitioning scheme in which the data set is repeatedly divided until the desired number of classes is reached. The latter scheme has been extended to mutliclass versions. In [15], a multiclass algorithm for the transductive learning problem in high-dimensional data classification, based on ℓ^1 relaxation of the Cheeger cut and the piecewise constant Mumford-Shah or Potts models, is described.

Our methods, on the other hand, have roots in the continuous setting as they are derived via a variational formulation. Alternative variational principles have also been used for image segmentation. In [73], a multiclass labeling for image analysis is carried out by a multidimensional total variation formulation involving a simplex-constrained convex optimization. In that work, a discretization of the resulting PDEs is used to solve numerically the minimization of the energy. A convex relaxation procedure is proposed and applied to image segmentation. In these cases, the discretization corresponds to a uniform grid embedded in the Euclidean space where the domain resides. Similarly, diffuse interface methods have been used successfully in image impainting [8,32] and image segmentation [39].

While our algorithms are inspired by continuous processes, they can be written directly in a discrete combinatorial setting defined by the graph Laplacian. This has the advantage, noted by Grady [57], of avoiding errors that could arise from a discretization process. We represent the data as nodes in a weighted graph, with each edge assigned a measure of similarity between the vertices it is connecting. The edges between nodes in the graph are not the result of a regular grid embedded in an Euclidean space. Therefore, a nonlocal calculus formulation [52] is the tool used to generalize the continuous formulation to a (nonlocal) discrete setting given by functions on graphs. Other nonlocal formulations for weighted graphs are included in [36], while [58] constitutes a comprehensive reference about techniques to cast continuous PDEs in graph form.The approach of defining functions with domains corresponding to nodes in a graph has successfully been used in areas, such as spectral graph theory [24, 84].

As pointed out in [9], there are interesting connections between the GL functional on graphs and normalized graph cuts. Shi and Malik [96] pose the problem of image segmentation as the solution of a generalized eigensystem generated from a graph Laplacian. In [10], graph cuts are used to efficiently find local minima of a wide class of energies with various smoothness constraints for multiclass image restoration. Also, as mentioned earlier, the method in [100] is a recursive graph-based partition scheme. A multiclass algorithm for the transductive learning problem in high-dimensional data classification, described in [15], is based on ℓ^1 relaxation of the Cheeger cut and the piecewise constant Mumford-Shah or Potts models. In [12], rigorous convergence results are presented for two algorithms that solve the relaxed Cheeger cut minimization used for unsupervised data clustering are presented. Our proposed methods are related to some of these approaches, but use the graph Ginzburg-Landau functional framework.

In the continuous setting, it can be shown that the GL is a diffuse interface approximation to the total variational functional [33, 70], and analogous results have recently been proved in the graph setting as well [103]. This function is a natural framework for producing smooth labels everywhere while preserving sharp discontinuities, with the sharpness controlled by a diffuse interface parameter. The advantage of the diffuse interface model is that the energy functional is more tractable, and can be minimized by simpler numerical methods.

3.4 Application to Image Processing

Below, we show the image processing results. To embed an image into a graphical framework, we consider each pixel as a vertex.

Application to Image Labeling

We applied our algorithm to segment objects in images of cows from the Microsoft image database. The goal was image labeling, where two images are inputted into the algorithm, one of which has been hand-segmented (partially or completely) into classes. The algorithm segments the second image based on the segmentation of the first.

Binary Image Labeling A fully connected graph is constructed, and the entries in the weight matrix are calculated using feature vectors. Every pixel in the image is assigned a feature vector consisting of intensity values of pixels in its neighborhood, which was of size 7×7 in our tests. We use the formula $w(x,y) = e^{-\frac{d(x,y)^2}{\sigma^2}}$, where $d(x,y)$ is the weighted 2-norm of the difference of the feature vectors of pixels x and y, and we add along the three RGB channels of the image. The weighted 2-norm modifies the components of the entered vector by giving more weight to the pixels close to the original pixel. We use a linearly decreasing kernel. This construction can be used to segment different types of objects using, for example, their color and texture features. Note that the weight function can be modified according to the image. For example, a weight function calculated using the spectral angle may be more effective in the segmentation of hyperspectral images.

To obtain eigenvalues and eigenvectors of $\mathbf{L}_s$, the Nyström extension method is used, since the size of the graph is large. For the problem, the fidelity term is the hand-labeled image, and we initialize u to be the class number if it's known and a middle value otherwise.

The results are displayed in Figure 3, where it is shown that our algorithm is robust to mislabeling in the hand labeled image. To transfer the label for the grass, cows and sky, our method needed about 29, 29, 27 seconds, respectively. The number of iterations in the minimization procedure (step 4 of the algorithm) and minimization time as compared to the method in [9] are displayed in Table 1. The calculations show that our method significantly reduces both.

Multiclass Image Labeling We also conducted the image labeling task using multiple classes. The results are shown in Figure 4. The weight matrix is constructed similarly to the way of the previous section, with the neighborhood of size 5×5. However, here we use the weight funtion (2) and create a sparse graph. A local scaling graph with $M = 30$ is constructed. For the fidelity term, 2.6% of labeled pixels are used.

The multiclass Ginzburg-Landau method used the following parameters: 30 eigenvectors, $\epsilon =$

	Minimization time in method in [9]	Minimization time in our method
grass label	8 s	3.5 s
cow label	18 s	3.5 s
sky label	6 s	1.8 s

	# of iterations in method in [9]	# of iterations in our method
grass label	130	22
cow label	274	29
sky label	84	11

Table 1: Comparison of Minimization Time and Number of Iterations of the MBO method and [9]

1, $dt = 0.1$, $\mu = 50$ and $\eta = 10^{-7}$. The average time for segmentation using different fidelity sets was 19.9 s. The multiclass MBO method used the following parameters: 30 eigenvectors, $dt = 0.1$, $\mu = 50$, $\eta = 10^{-7}$. The average time for segmentation over 10 runs was around 1.2 s.

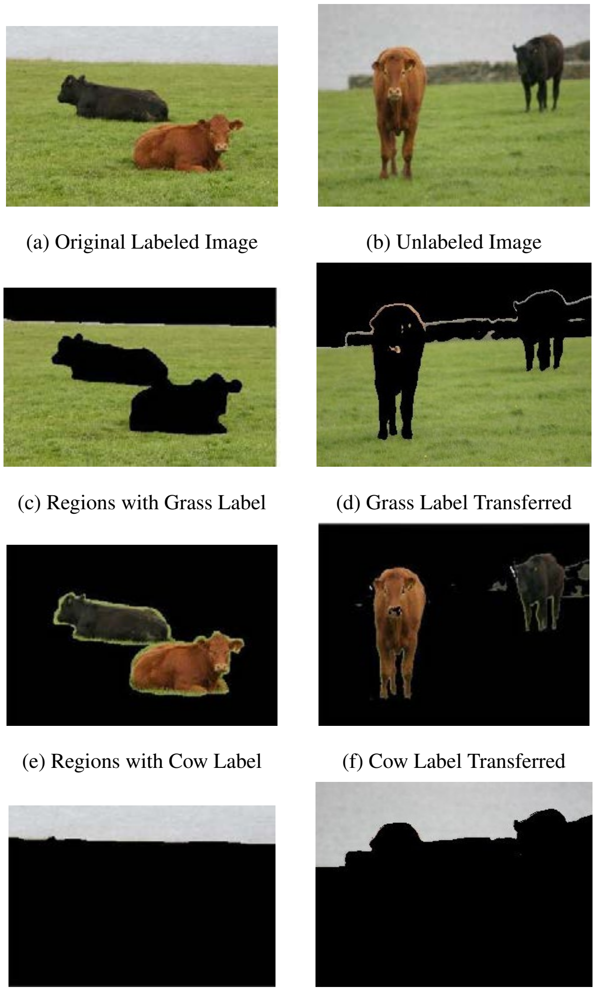
Application to Image Segmentation

Grayscale Image Segmentation We tested our algorithms on the image of figures shown in Figure 5a. This is a 191×196 pixel grayscale image, to be divided into five classes. To construct the weight matrix, we use feature vectors corresponding to a pixel's x-coordinate, y-coordinate, and intensity. A local scaling graph with $M = 30$ is constructed. For the fidelity term, 1,500 or 4% of the points were randomly chosen.

The original image as well as the segmentation results are included in Figure 5. In each segmentation, the white regions correspond to the identified class. The multiclass Ginzburg-Landau method used the following parameters: 10 eigenvectors, $\epsilon = 1$, $dt = 0.5$, $\mu = 50$ and $\eta = 10^{-7}$. It was able to segment the classes perfectly, with an average time of 4.1 s. The multiclass MBO method used the following parameters: 10 eigenvectors, $dt = 0.5$, $\mu = 50$, $\eta = 10^{-7}$. The algorithm was able to segment all the points correctly in 2 iterations and 0.232 s.

We compare our result to the method of Li and Kim [76], which also segments the image perfectly. However, their method requires additional information, such as the densities of each class, that we do not need in our method.

Multiclass Image Segmentation We then tested our algorithms on the color image of cows



(a) Original Labeled Image

(b) Unlabeled Image

(c) Regions with Grass Label

(d) Grass Label Transferred

(e) Regions with Cow Label

(f) Cow Label Transferred

(g) Regions with Sky Label

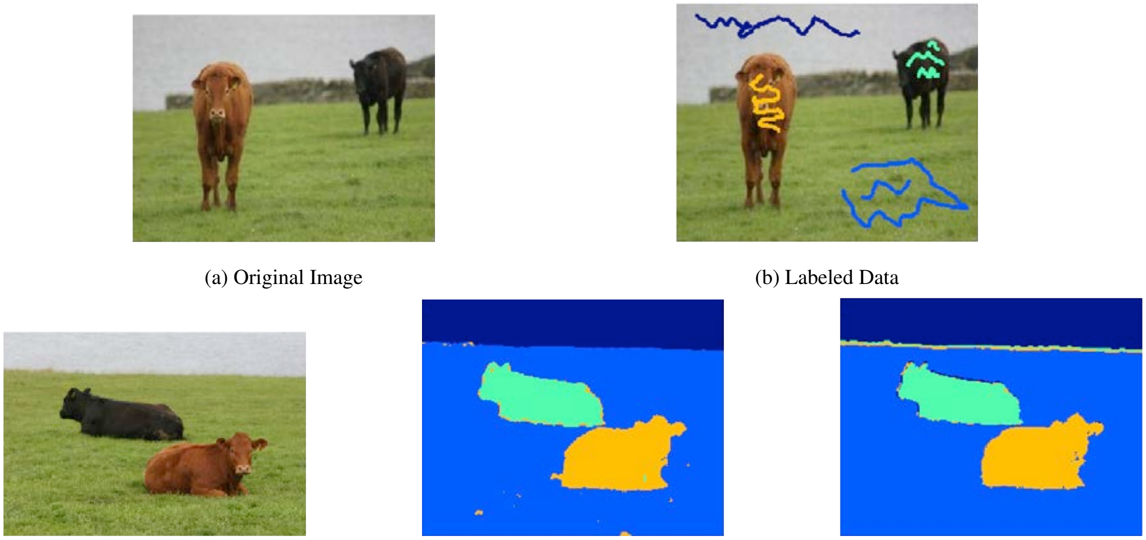
(h) Sky Label Transferred

Figure 3: Image Labeling Example 1

shown in Figure 6a. This is a 213×320 color image, to be divided into four classes: sky, grass, black cow and red cow. The graph is constructed in the same way as in the multiclass image labeling example. For the fidelity term, 2.6% of labeled pixels are used.

The multiclass GL method used the following parameters: 30 eigenvectors, $\epsilon = 1$, $dt = 0.1$, $\mu = 50$ and $\eta = 10^{-7}$. The average time for segmentation using different fidelity sets was 19.9 s. The multiclass MBO method used the following parameters: 30 eigenvectors, $dt = 0.1$, $\mu = 50$, $\eta = 10^{-7}$. The average time for segmentation was around 1.2 s, and there were 11 iterations.

One of the results of each of our two methods (using the same fidelity set) is depicted in Figure 6. It can be seen that both methods are able to identify all the classes.



(a) Original Image

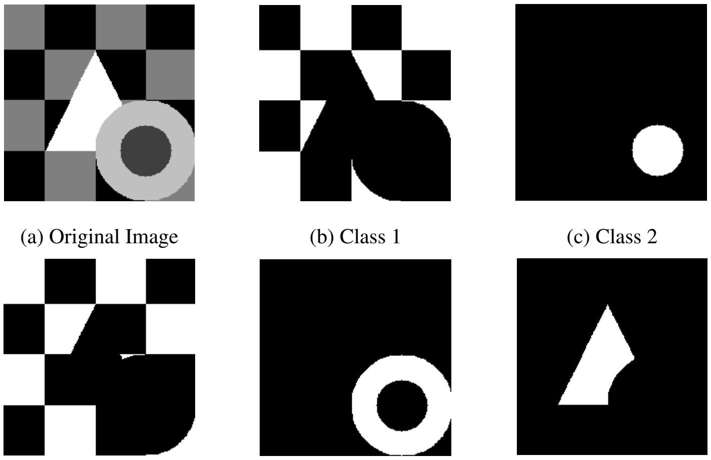
(b) Labeled Data

(c) Image to Segment

(d) Multiclass GL

(e) Multiclass MBO

Figure 4: Image Labeling Example 2



(a) Original Image

(b) Class 1

(c) Class 2

(d) Class 3

(e) Class 4

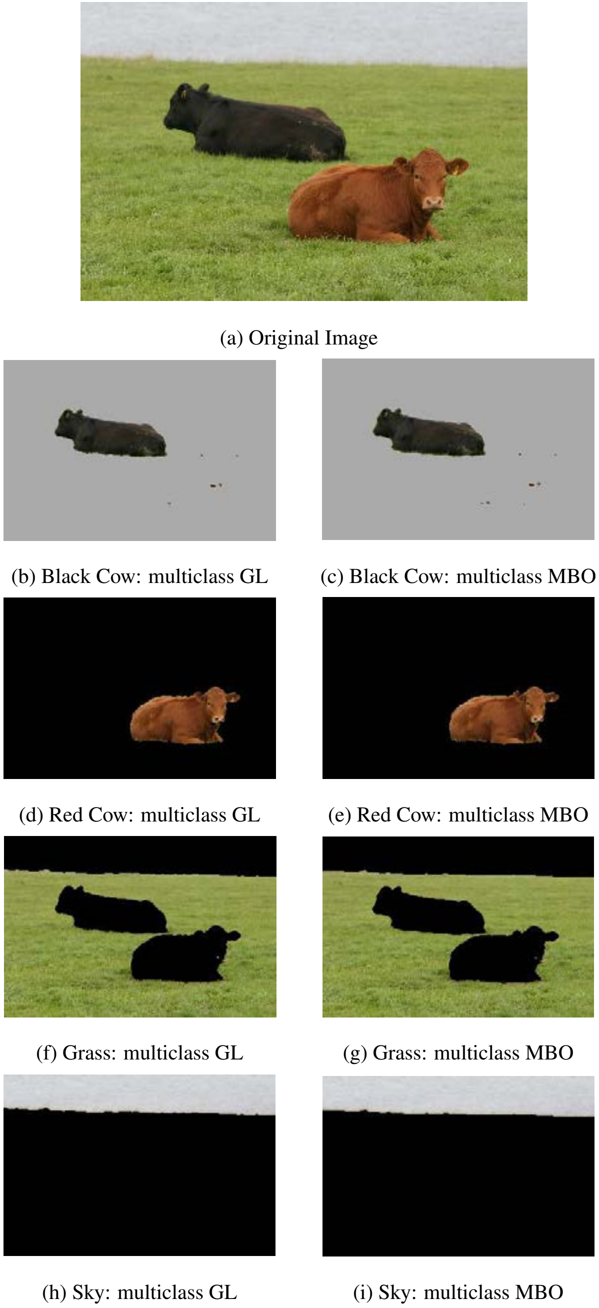
(f) Class 5

Figure 5: Image Segmentation Example 1

Application to Image Inpainting

The problem of fitting information in the missing pixels of an image is an important inverse problem in image processing with various applications. Obviously, the goal is to produce a modified image that will look natural to an observer. The problem of inpainting may also be seen as the problem of removing occlusive objects from an image. Sparse reconstruction refers to the problem of recovering randomly distributed missing pixels.

There are numerous approaches to solve these problems in the current literature. Local TV



(a) Original Image

(b) Black Cow: multiclass GL

(c) Black Cow: multiclass MBO

(d) Red Cow: multiclass GL

(e) Red Cow: multiclass MBO

(f) Grass: multiclass GL

(g) Grass: multiclass MBO

(h) Sky: multiclass GL

(i) Sky: multiclass MBO

Figure 6: Image Segmentation Example 2

methods became state-of-the-art techniques for image impainting. However, since they do not perform well on images with high texture, methods that decompose images into cartoon and texture and simultaneously inpaint both are developed [7, 94]. The problem is also solved with nonlocal inpainting methods. We are particularly interested in the nonlocal inpainting algorithm from [51] as we develop a computationally efficient nonlocal method. Some very successful nonlocal methods for inpainting and sparse reconstruction are given in [3] and [42]. Recently, the class of methods that use dictionaries of small patches that commonly appear in natural images became increasingly popular. Those methods, besides inpainting, are also successful in denoising as shown in [78]. In addition, a method for image inpainting using Navier-Stokes fluid dynamics is proposed in [9]. The authors use Navier-Stokes dynamics to propagate isophotes into the inpainting region, thus simulating the way painting restoration is done. Wavelets and framelets are also successfully applied to solve inpainting problems [19, 32].

We modify our segmentation algorithm slightly for the purpose of image inpainting.

Binary Image Inpainting Although the key steps of the segmentation algorithm remain the same when it is modified for image inpainting, there are differences to be noted. For example, if a damaged image is used to construct the adjacency matrix W, the results might not be accurate, so we first apply a fast and simple H^1 inpainting algorithm on the image and then use the result to create W. The H^1 inpainting algorithm we apply is just the local minimization problem:

$$\min_u \int |\nabla u|^2 dx + \int \lambda(x)(u - u_0)^2 dx.$$

The latter term in the sum is the fidelity term, and, of course, we are not limited to this formulation of it. Although the latter algorithm is fast, it does not perform well on images with high textures and repetitive structures nor does it preserve edges [46], something that we achieve.

The matrix W is built by using a window of a certain size around each pixel. We set $W(x, y) = 0$ for all pixels j that are not in the window of pixel i. Inside the window, $W(x, y) = w(x, y)$, where the weight function is calculated in the same way as for binary image labeling. No updating of the matrix W is necessary. The Rayleigh-Chebyshev procedure is used to calculate the eigenvectors and eigenvalues of the graph Laplacian for binary inpainting. The fidelity region is the non-damaged region. We initialize u to be the middle value for the non-fidelity points.

Grayscale Image Inpainting To generalize to graycale inpaining, we split the signal bit-wise into channels, as in [32]:

$$u(x) = \sum_{m=0}^{K-1} u_m(x) 2^m,$$

where u_m denotes the m^{th} digit in the binary representation of the signal, and $u_m \in \{0, 1\}$ for $\forall x$.

A fully connected graph is created in the same way as in the binary inpainting case. The Nyström extension method is used to calculate the eigenvalues and corresponding eigenvectors since the size of the graph is very large. The fidelity region and initialization is the same as in the binary inpainting case.

Updating the weight matrix is often necessary for grayscale inpainting, since the adjacency matrix formed from the damaged image is usually not good enough to restore texture and complex patterns, as it contains “bad” regions whose values lie far from the true value. In our tests, every few iterations, the matrix is updated using the result from the last iteration as the “new image”.

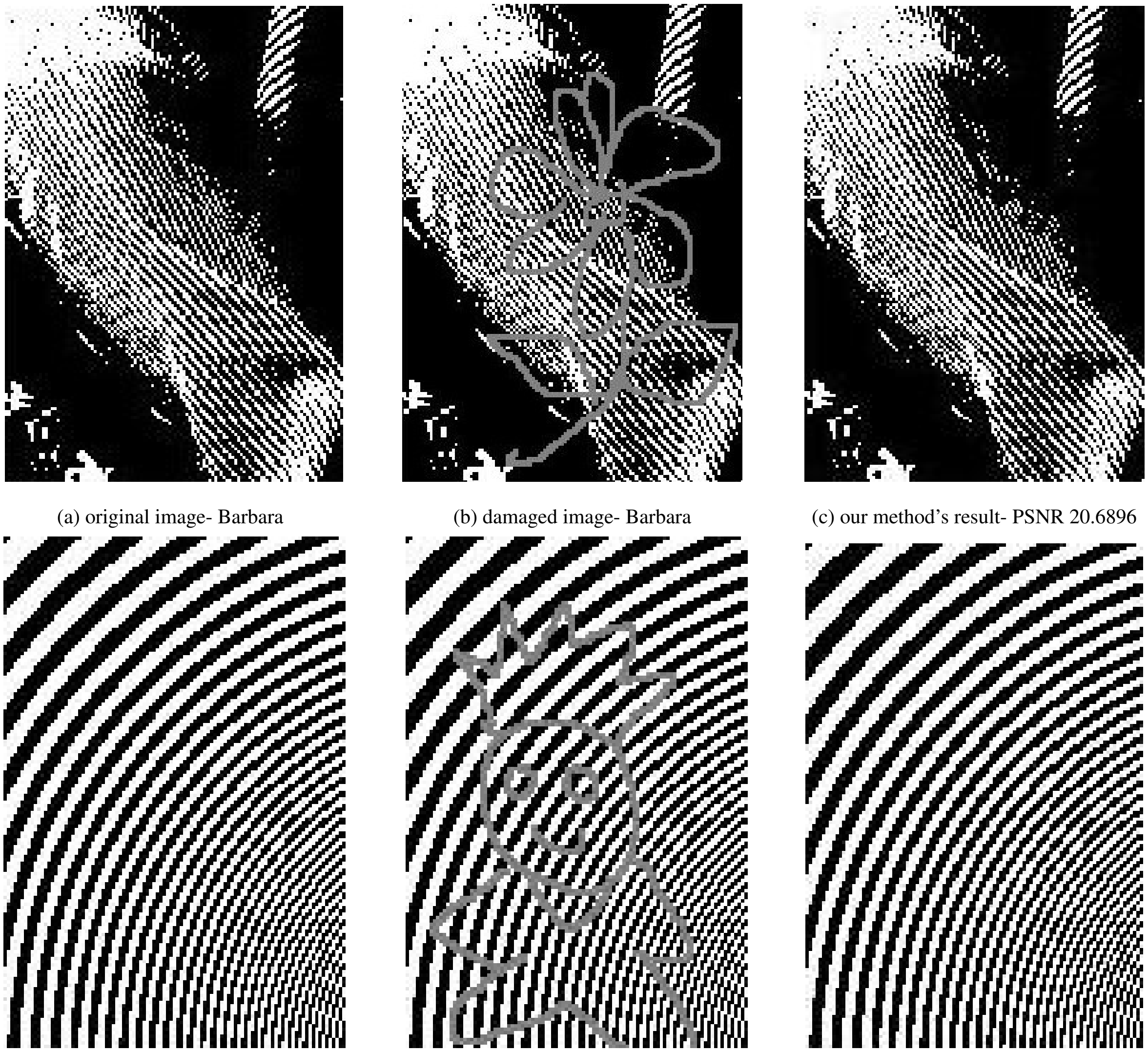
Binary Image Inpainting Results The binary image inpainting results and their PSNR are displayed in Figure 7. In both cases, the algorithm was able to recover the texture and repetitive structure present in the image.

Grayscale Image Inpainting Results The grayscale inpainting results along with their PSNR are displayed in Figures 8-13. In all cases, repetitive structure and texture were recovered.

We compare our results to local and nonlocal TV inpainting. Local TV inpainting fails to recover texture and repetitive structure. While the results of nonlocal TV inpainting are comparable to those of our method, our method is more efficient. Timing results are displayed in Table 2, and we see that our method is several times faster. We also show our method and nonlocal TV inpainting at certain iterations in Figure 14. To implement the nonlocal TV inpainting algorithm, we used the Bregmanized version detailed in [112] and modified it for inpainting. The stopping condition was the same as in our inpainting algorithm, and a quick H^1 inpainting algorithm was run on the image before the weights were calculated.

	Total time for nonlocal TV	Total time for our method
chessboard-like pattern	266 s	48 s
text inpainting	410 s	67 s
small rectangle inpainting	1882 s	443 s
large rectangle inpainting	3397 s	832 s
50% inpainting	1402 s	333 s

Table 2: Comparison of Runtime of the MBO method and that of Nonlocal TV



(a) original image- Barbara

(b) damaged image- Barbara

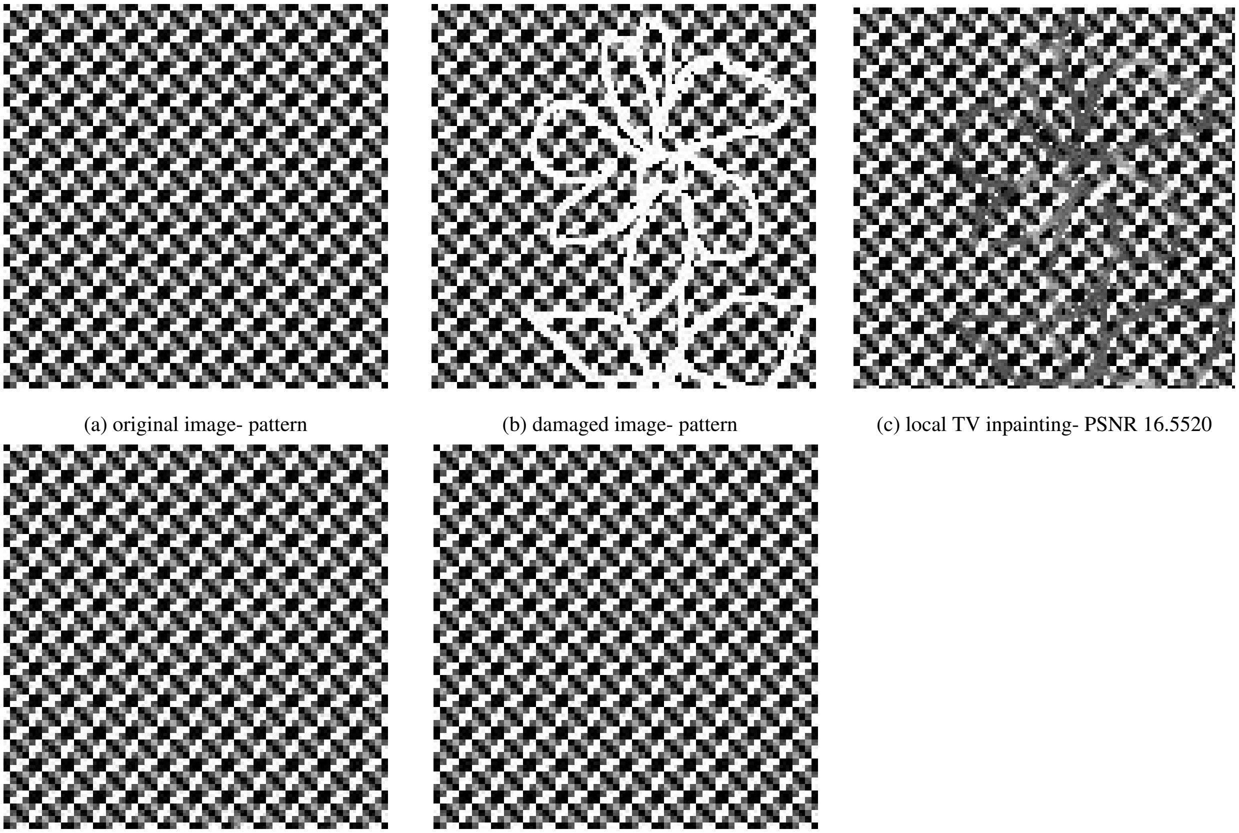
(c) our method's result- PSNR 20.6896

(d) original image- stripes

(e) damaged image- stripes

(f) our method's result- PSNR 25.0687

Figure 7: Binary Inpainting Example. For the Barbara image, the simulation took 113 seconds, and there were 6 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.003$, $\sigma = 45$, 31×31 neighborhood for feature vector calculation, 21×21 window and calculated 400 eigenvectors. For the image of stripes, the simulation took 66 seconds, and there were 4 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.002$, $\sigma = 45$, 17×17 neighborhood for feature vector calculation, 21×21 window and calculated 200 eigenvectors.



(a) original image- pattern

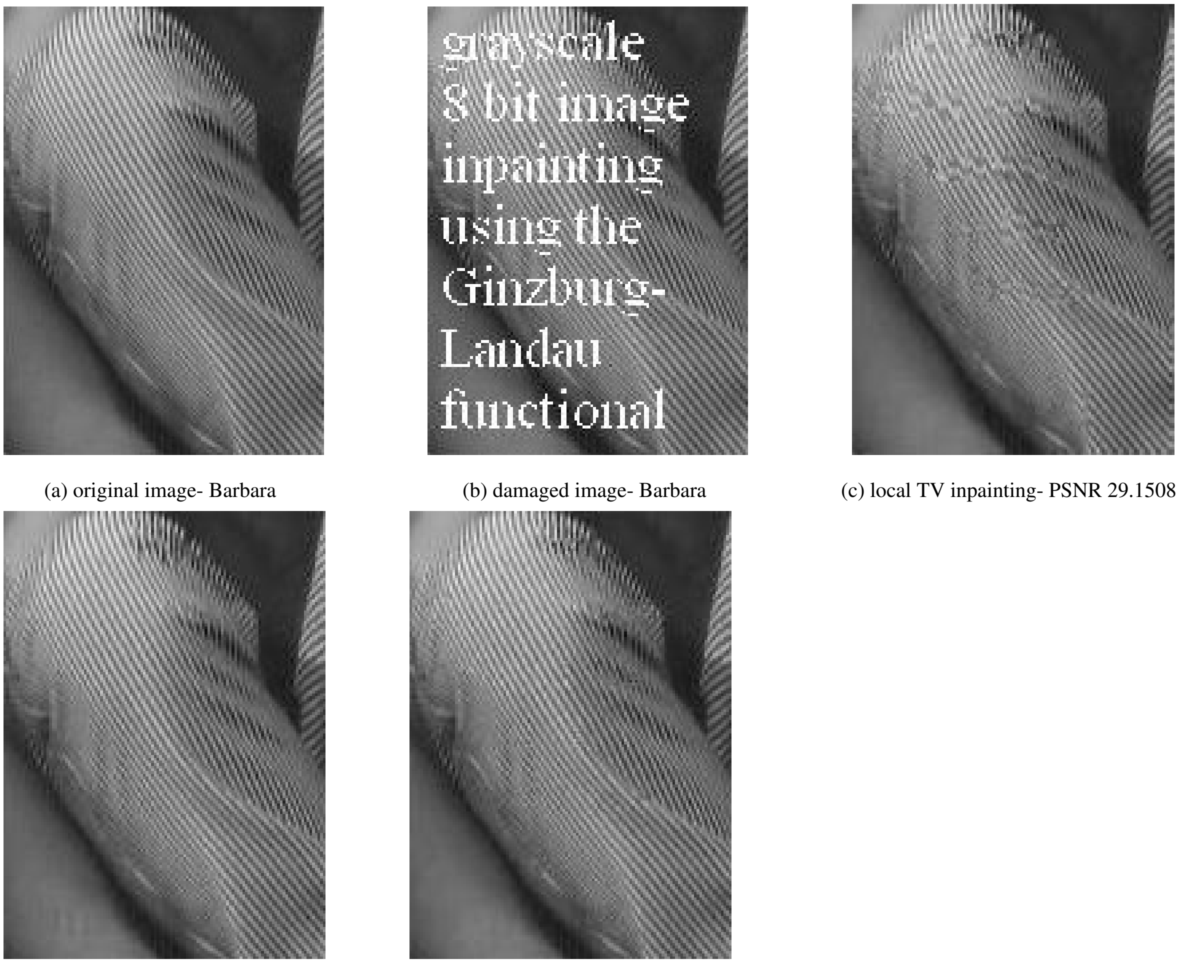
(b) damaged image- pattern

(c) local TV inpainting- PSNR 16.5520

(d) nonlocal TV inpainting- PSNR 41.3891

(e) our method's result- perfect reconstruction

Figure 8: Grayscale Inpainting Example. The simulation took 48 seconds, and there were 2 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 20$, 41×41 neighborhood for feature vector calculation, and calculated 600 eigenvectors. No updating of W was necessary. The nonlocal inpainting took 266 seconds.



(a) original image- Barbara

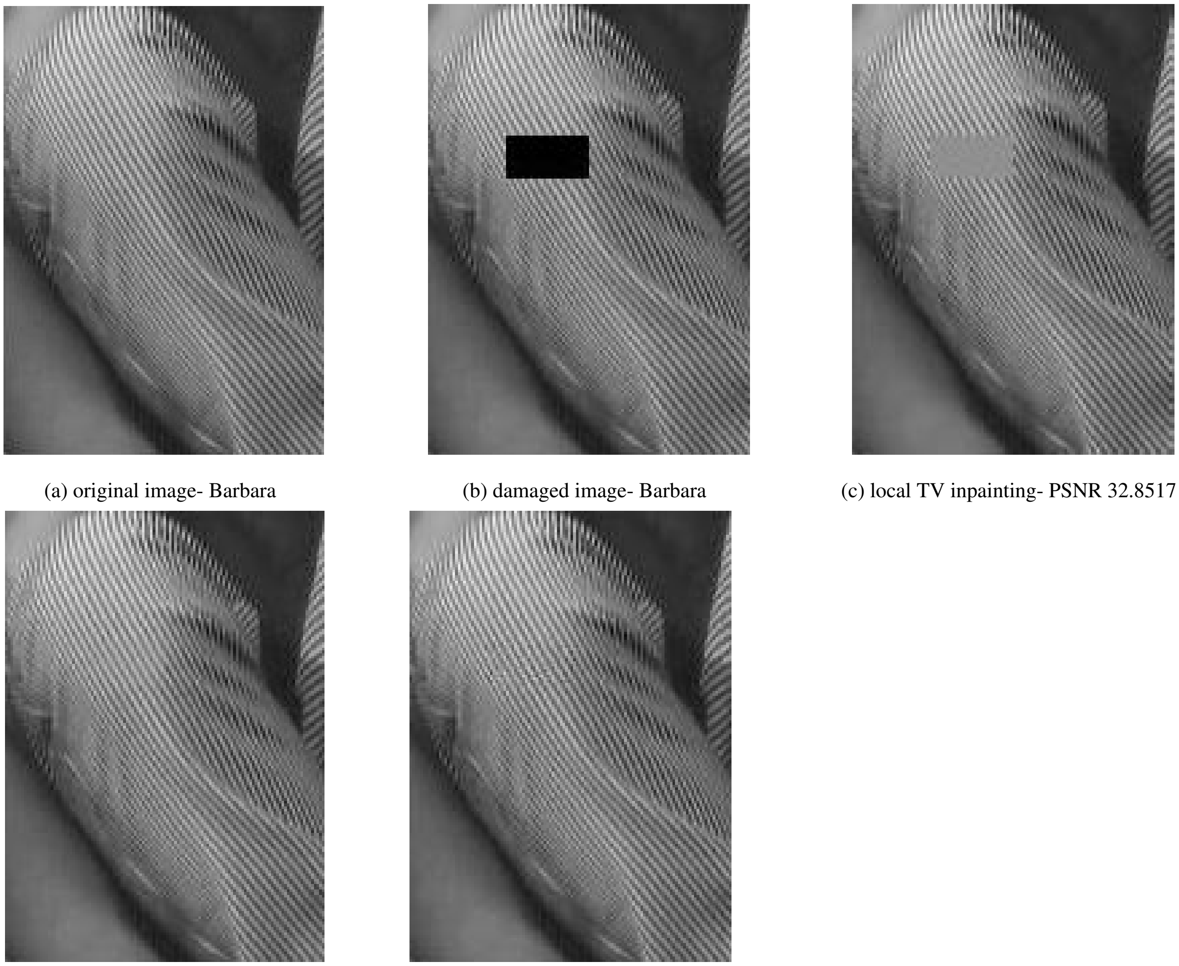
(b) damaged image- Barbara

(c) local TV inpainting- PSNR 29.1508

(d) nonlocal TV inpainting- PSNR 35.6896

(e) our method's result- PSNR 34.0688

Figure 9: Text Inpainting Example. The simulation took 67 seconds, and there were 4 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 5$, 21×21 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every other iteration. The nonlocal inpainting took 410 seconds.



(a) original image- Barbara

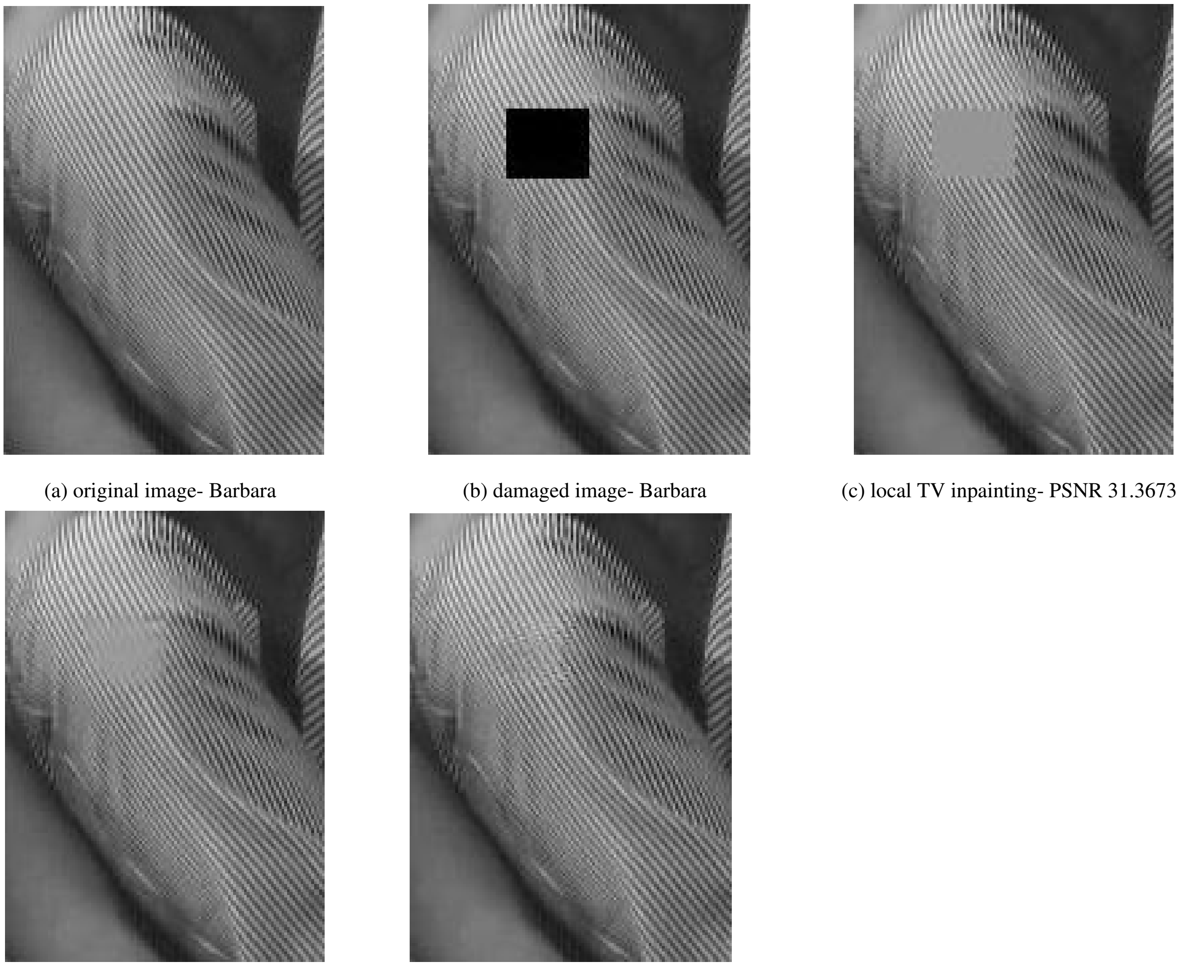
(b) damaged image- Barbara

(c) local TV inpainting- PSNR 32.8517

(d) nonlocal TV inpainting- PSNR 44.1469

(e) our method's result- PSNR 41.2848

Figure 10: Region Inpainting Example 1. The simulation took 443 seconds, and there were 13 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.01$, $\sigma = 4$, 31×31 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every iteration. The nonlocal TV inpainting took 1882 seconds.



(a) original image- Barbara

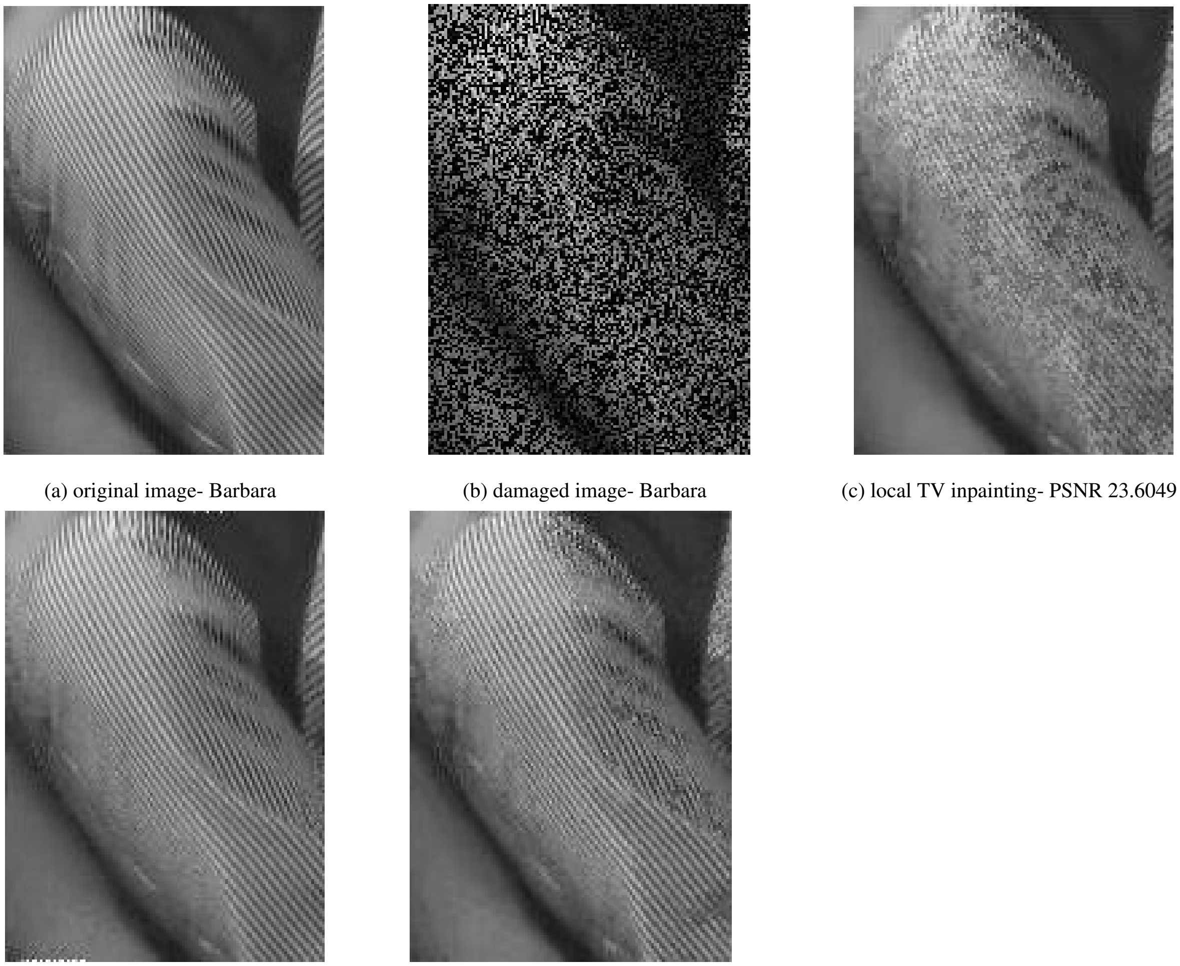
(b) damaged image- Barbara

(c) local TV inpainting- PSNR 31.3673

(d) nonlocal TV inpainting- PSNR 35.0663

(e) our method's result- PSNR 37.0315

Figure 11: Region Inpainting Example 2. The simulation took 832 seconds, and there were 13 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.014$, $\sigma = 4$, 45×45 neighborhood for feature vector calculation, and calculated 500 eigenvectors. We update W every iteration. The nonlocal inpainting took 3397 seconds.



(a) original image- Barbara

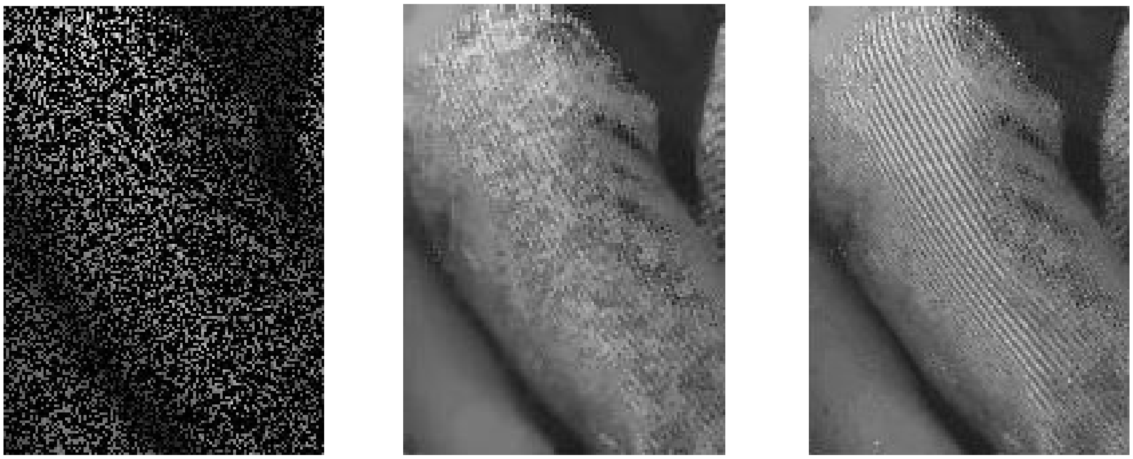
(b) damaged image- Barbara

(c) local TV inpainting- PSNR 23.6049

(d) nonlocal TV inpainting- PSNR 27.8196

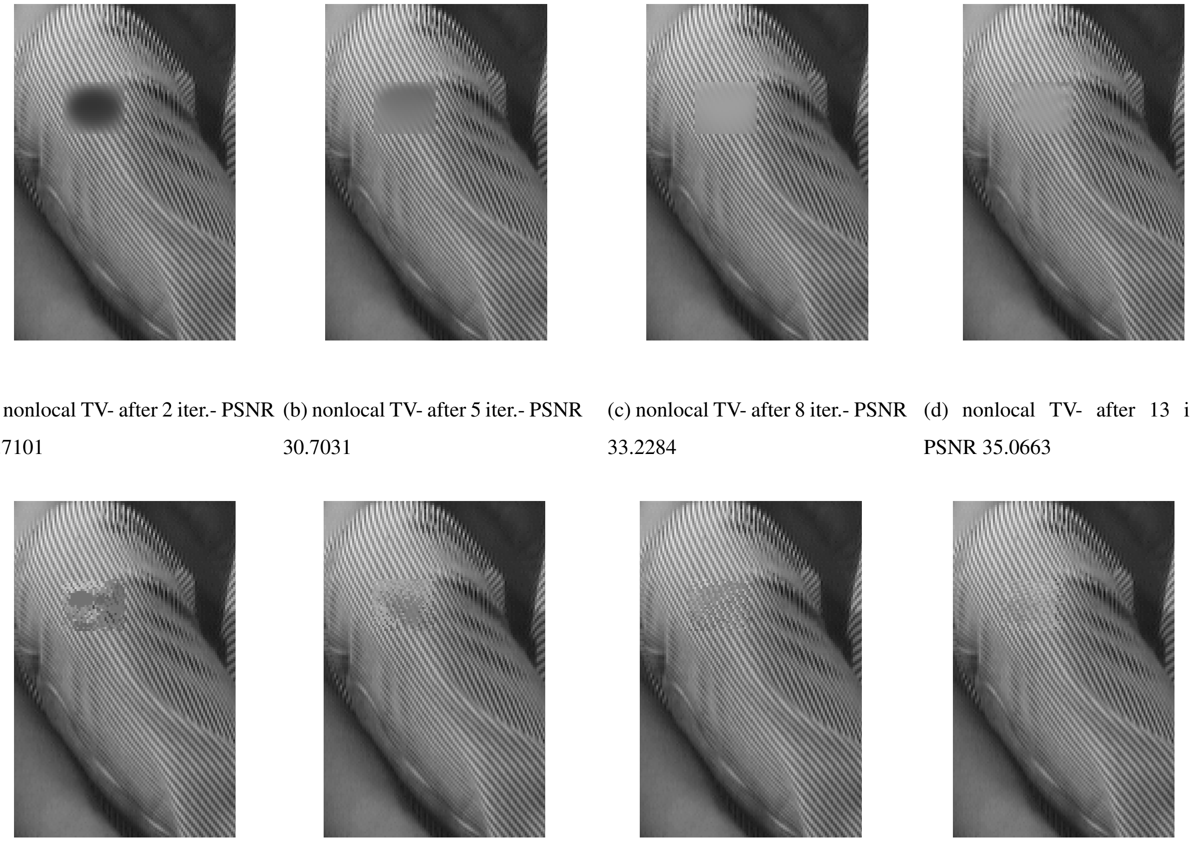
(e) our method's result- PSNR 27.1651

Figure 12: 50% Reconstruction Example. The simulation took 333 seconds, and there were 50 iterations in the two-step scheme. We used $C_1 = 700$, $dt = 0.005$, $\sigma = 4$, 7×7 neighborhood for feature vector calculation, and calculated 400 eigenvectors. We update W every iteration. The nonlocal inpainting took 1402 seconds.



(a) damaged image- 35% of the pixels removed (b) local TV inpainting- PSNR 22.6530 (c) our method's result- PSNR 24.1266

Figure 13: 65% Reconstruction Example



(a) nonlocal TV- after 2 iter.- PSNR 25.7101

(b) nonlocal TV- after 5 iter.- PSNR 30.7031

(c) nonlocal TV- after 8 iter.- PSNR 33.2284

(d) nonlocal TV- after 13 iter.- PSNR 35.0663

(e) our method- after 2 iter.- PSNR 30.4406

(f) our method- after 5 iter.- PSNR 31.8993

(g) our method- after 8 iter.- PSNR 34.4851

(h) our method- after 13 iter.- PSNR 37.0315

Figure 14: Visualization of Inpainting at Different Iterations

3.5 Application to Classification

We have applied our algorithm to both the binary and multiclass classification problem and tested it on benchmark data sets.

Application to Binary Classification

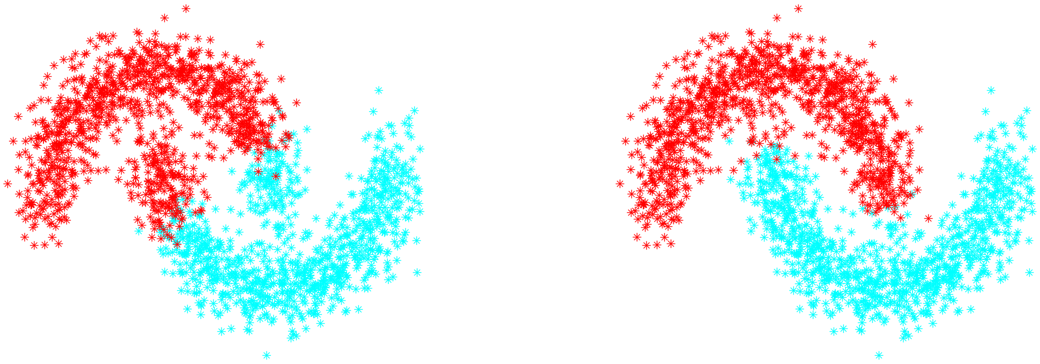
Two Moons Data Set This data set was used in [18] in relation to spectral clusering. It is constructed from the following two half circles in $\mathbb{R}^2$ with radius one. The first half circle is centered at the origin and is in the upper half plane. The second half circle is formed by taking the lower half of the circle centered at $(1, 0.5)$. A thousand points are chosen uniformly from each of the two half circles. The two thousand points are then embedded in $\mathbb{R}^{100}$, and i.d.d. Gaussian noise with standard deviation 0.02 is added to each coordinate. The goal is to segment those two half circles.

To create the graph, we used the weight function (2), and thus created a sparce graph. To calculate the eigenvectors, the Rayleigh-Chebyshev procedure [1] is used. There is no fidelity term, but we use a zero mass constraint due to the equal size of the classes. For initialization of u, we use the sign of the second eigenvector of the symmetric Laplacian after the mean constraint has been applied to it (see Section 15.4).

We compared our results to the method of Bertozzi and Flenner in [9] by running simulations on 35 different randomly generated two moons data sets. The average accuracy was 96.0520% and 96.0460% for our method and the method in [9], respectively. However, 40 iterations in the minimization procedure were used, compared to 300 needed using the method in [9]. Therefore, our method resulted in a significant decrease in the number of iterations.

We also compared our results to a spectral clustering method of thresholding the second eigenvector of L_s. The results are displayed in Figure 15. Clearly, clustering using the second eigenvector does not result in an accurate segmentation.

House Voting Records Data Set We applied our algorithm to the US House of Representatives voting records data set, which consists of 16 different votes from each of the 435 individuals. The goal was to assign each individual to either the Republican or the Democrat party using the prior knowledge of the party affiliation of only five individuals, two Democrats and three Republicans. The votes were taken in 1984 from the 98th United States Congress, 2nd session.



(a) second eigenvector segmentation- 83.75%

(b) our method's segmentation- 97.7%

Figure 15: Two Moons Data Set Results. Segmentation by thresholding the second eigenvector and our method, respectively. The four parameters s (in step IV of our algorithm), number of eigenvectors, dt, and M (parameter in the Zelnik-Manor and Perona weight function) are set to 3, 25, 0.725 and 13, respectively.

A weight matrix is constructed using calculations involving feature vectors. A 16-dimensional feature vector is assigned to each individual consisting of his/her 16 votes. A "yes" vote is set to 1, a "no" vote is set to -1, while a "did not vote" recording is set to 0. The weight function used is (1). The graph is made sparse by setting $W(x, y)$ equal to zero if point y is not among the M^{th} closest points to point x. To calculate the eigenvectors, a SVD solver is used. We initialize u to the middle value for non-fidelity points. The fidelity term consisted of two Democracts and three Republicans. The three Republicans were chosen to be the first, second and eighth person in the list. The Democrats were chosen to be the third and fourth person in the list. The parameters C_1 (fidelity term parameter), s (in step IV of our algorithm), number of eigenvectors, dt, σ and M are set to 9.25, 3, 45, 4.675, $\sqrt{5}$ and 10, respectively.

We obtained an accuracy of 94.023%. Only 5 iterations in the minimization procedure were needed compared to 450 iterations needed by the method in [9]. Some of the votes predicted the party affiliation very well, *i.e.* above 85%. We investigated the accuracy of our algorithm when these votes were removed. With top two, top six and top eight most predictive votes removed, our method obtained an accuracy of 90.1149%, 88.34448% and 81.1494%, resp.. The order of the top eight predictive votes from the most predictive to least predictive is vote 4, 14, 1, 2, 15, 6, 3 and 8.

Application to Multiclass Classification

For multiclass classification, we have tested our algorithms on several benchmark data sets. In all cases, we compute the graph by (2), and make it sparse by only connecting vertices that are near enough to each other.

All the results and comparisons with other published methods are summarized in Tables 14 and 4. Due to the arbitrary selection of the fidelity points, our reported values correspond to averages obtained over 10 runs with different random selections. The timing results and number of iterations of the two methods are shown in Tables 5 and 6, respectively. The methods are labeled as "multiclass GL" and "multiclass MBO". These comparisons show that our methods exhibit a performance that is competitive with or better than the current state-of-the-art segmentation algorithms.

Parameters are chosen to be compatible between the methods. For the multiclass GL method, the convexity constant used is: $C = \mu + \frac{1}{\epsilon}$. This is the minimum constant that guarantees the convexity and concavity of the terms in the energy partition of the convex splitting strategy employed. For the multiclass MBO method, as discussed in the previous section, the diffusion step can be repeated a number of times before thresholding. In all of our results, we run the diffusion step three times before any thresholding is done ($N_S = 3$). To compute the eigenvectors and eigenvalues of the symmetric graph Laplacian, we use fast numerical solvers. We use the Rayleigh-Chebyshev procedure of [1] for computing all the eigendecompositions.

All the results reported point out that both multiclass GL and multiclass MBO perform well in terms of data segmentation accuracy. While the ability to tune multiclass GL can be an advantage, multiclass MBO is simpler and, in our examples, displays even better performance in terms of its greater accuracy and tiny number of iterations required. The relative strength and speed of multiclass MBO may not always hold, but the avoidance of a nonconvex functional minimization that takes place in multiclass GL may explain the accuracy and speed increase. Exploring the underlying connections of the energy evolution of these methods and the energy landscape for the relaxed Cheeger cut minimization recently established in [12] are to be explored in future work.

Synthetic Data The synthetic data set we used is the three moons data set. It is constructed by generating three half circles in $\mathbb{R}^2$. The two half top circles are unit circles with centers at $(0, 0)$ and $(3, 0)$. The bottom half circle has radius 1.5 and the center at $(1.5, 0.4)$. Five hundred points from each of those three half circles are sampled and embedded in $\mathbb{R}^{100}$ by adding Gaussian noise with standard deviation of 0.14 to each of the 100 components of each point. The dimensionality of the data set and the noise makes segmentation a significant challenge.

The weight matrix of the graph edges was calculated using local scaling based on the 17^{th} closest point ($M = 17$). The fidelity term was constructed by labeling 25 points per class, 75 points in total, corresponding to only 5% of the points in the data set. The multiclass GL method used the following parameters: 20 eigenvectors, $\epsilon = 1$, $dt = 0.1$, $\mu = 30$, $\eta = 10^{-7}$. The method was able to produce an average of 98.4% of correct classification, with a corresponding computation time of 0.16 s per run on a 2.4 GHz Intel Core i2 Quad without any parallel processing. The multiclass MBO method used the following parameters: 20 eigenvectors, $dt = 0.1$, $\mu = 30$, $\eta = 10^{-7}$. It was able to segment an average of 99.12% of the points correctly over 10 runs with only 3 iterations and about 0.01 s of computation time. One of the results obtained is shown in Figure 16.

Table 14 gives published results from other related methods, for comparison. Note that the results for p-Laplacians [18], Cheeger cuts [100] and binary GL are for the simpler binary problem of two moons (also embedded in $\mathbb{R}^{100}$). While, strictly speaking, these are unsupervised methods, they all incorporate prior knowledge such as a mass balance constraint. We therefore consider them comparable to our SSL approach. The "tree GL" method [47] uses a scalar multiclass GL approach with a tree metric. It can be seen that our methods achieve the highest accuracy on this test problem.

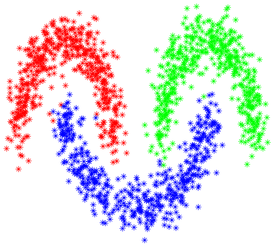


Figure 16: Three Moons Data Set Result. Segmentation of three moons using multiclass MBO (98.4667% correct).

MNIST Data Set The MNIST data set [72] is composed of $70,000$ 28×28 images of handwritten digits 0 through 9. Examples of entries can be found in Figure 17. The task is to classify each of the images into the corresponding digit. The images include digits from 0 to 9; thus, this is a 10 class segmentation problem.

To construct the weight matrix, we used the local scaling based on the 8^{th} closest neighbor

($M = 8$). Note that we perform no preprocessing, i.e. the graph is constructed using the 28×28 images. For the fidelity term, 250 images per class (2500 images corresponding to 3.6% of the data) are chosen randomly. The multiclass GL method used the following parameters: 300 eigenvectors, $\epsilon = 1$, $dt = 0.15$, $\mu = 50$ and $\eta = 10^{-7}$. The complete set of 70,000 images was segmented with an average accuracy of 96.8% of the digits classified correctly in an average time of 811 s. The averages are obtained over 10 runs. The confusion matrix for the best result obtained is included in Table 7. Most of the mistakes were in distinguishing digits 4 and 9, and digits 5 and 8. The multiclass MBO method used the following parameters: 300 eigenvectors, $dt = 0.15$, $\mu = 50$, $\eta = 10^{-7}$. The algorithm was able to segment an average of 96.91% of the digits correctly over 10 runs in only 4 iterations and 15.382 s. We display the confusion matrix in Table 8. Note that most of the mistakes were in distinguishing digits 4 and 9, and digits 2 and 7.

Table 14 compares our results with those from other methods in the literature. As with the three moon problem, some of these are based on unsupervised methods but incorporate enough prior information that they can fairly be compared with SSL methods. The methods of linear/nonlinear classifers, k-nearest neighbors, boosted stumps, neural and convolutional nets and SVM are all supervised learning approaches, taking 60,000 of the digits as a training set and 10,000 digits as a testing set [72], in comparison to our SSL approaches where we take only 3.6% of the points for the fidelity term. Our algorithms are nevertheless competitive with, and in most cases outperform, these supervised methods. Moreover, we perform no preprocessing or initial feature extraction on the image data, unlike most of the other methods we compare with (we did exclude from the comparison, however, methods that explicitly deskewed the image). While there is a computational price to be paid in forming the graph when data points use all 784 pixels as features (see graph calculation time in Table 5), this is a one-time operation that conceptually simplifies our approach.

Figure 17: Examples of Digits from the MNIST Data Set

COIL Data Set We evaluated our performance on the benchmark COIL data set [22, 88]. This is a set of color 128×128 images of 100 objects, taken at different angles. The red channel of each image was then downsampled to 16×16 pixels by averaging over blocks of 8×8 pixels. Then 24 of the objects were randomly selected and partitioned into six classes. Discarding 38 images from each class leaves 250 per class, giving a data set of 1500 data points.

To construct the weight matrix, we used a local scaling based on the 4^{th} closest neighbor ($M = 4$). The fidelity term was constructed by labeling 10% of the points, selected at random. For multiclass GL, the parameters were: 50 eigenvectors, $\epsilon = 1$, $dt = 0.2$, $\mu = 100$ and $\eta = 10^{-7}$. This resulted in 91.4% of the points classified correctly (average) in 2.3 s. For multiclass MBO, the parameters were: 50 eigenvectors, $dt = 0.2$, $\mu = 100$, $\eta = 10^{-7}$. We obtained an accuracy of 91.46%, averaged over 10 runs. The procedure took 6 iterations and 0.03 s.

Comparative results reported in [98] are shown in Table 14. These are all SSL methods (with the exception of k-nearest neighbors which is supervised), using 10% fidelity just as we do. Our results are of comparable or greater accuracy.

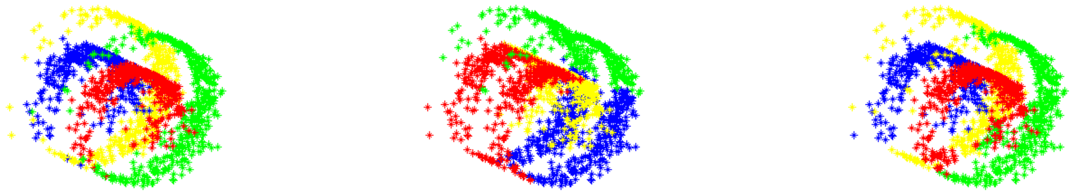
WebKB Data Set We tested our methods on the task of text classification on the WebKB data set [28]. This is a collection of 4199 webpages from Cornell, Texas, Washington and Wisconsin universities, as well as other miscellaneous pages from other universities. The webpages are to be divided into 4 classes: project, course, faculty and student. The data set is preprocessed as in [20].

To construct the weight matrix, we used 575 nearest neighbors. Tfidf term weighting [20] is used to represent the website feature vectors. They were then normalized to unitary length. The weight matrix points are calculated using cosine similarity. For the multiclass GL method, the parameters were: 250 eigenvectors, $\epsilon = 1$, $dt = 1$, $\mu = 50$ and $\eta = 10^{-7}$. The average accuracies obtained are: 81.5%, 84.2%, 85.4%, 86.7% and 87.3% over fidelity sets of 10%, 15%, 20%, 25% and 30% of the points, respectively. The average computation time is 6.97 s. For the multiclass MBO method, the parameters were: 250 eigenvectors, $dt = 1$, $\mu = 4$, $\eta = 10^{-7}$. We obtained average accuracies of: 83.71%, 85.75%, 86.81%, 87.74% and 88.48% over fidelity sets of 10%, 15%, 20%, 25% and 30% of the points, resp.. The procedure took 0.05 s and 7 iterations.

We compare our results with those of several supervised learning methods reported in [20], shown in Table 14. For these methods, two thirds of the data was used for training, and one third for testing. Our SSL methods obtain higher accuracy, using only 20% fidelity (for multiclass MBO). Note also that a larger sample of points for the fidelity term reduces the error in the classification results, as shown in Table 4. Nevertheless, the accuracy is high even for the smallest fidelity sets. Therefore, the methods appear quite adequate for the SSL setting where only a few labeled data points are known beforehand.

Swiss Roll Data Set The Swiss roll data set, pictured in Figure 18a, contains 1600 $3D$ points arranged in spirals. To calculate the weight matrix, we used the weight function in [90] with 10 nearest neighbors. To calculate the eigenvectors, we used the procedure in [1]. The parameters used were: 50 eigenvectors, $C = 51$, $\epsilon = 1$, $dt = 0.1$, $\mu = 50$ and $\eta = 10^{-7}$ with 80 fidelity points (5%).

Averaged over 100 runs, the average accuracies obtained were 94.1% and 91.8% for the multiclass GL and MBO methods, respectively. The visual result of the latter method is included in Figure 18c. We compare this result to the one obtained using spectral clustering (Figure 18b): average accuracy was only 49.75% over 100 runs, with the graph being the same as for multiclass GL and MBO case. Calculations were done with 4 eigenvectors.



(a) Swiss roll data set

(b) spectral clustering result (49%)

(c) our method's result (94.5%)

Figure 18: MBO Method Swiss Roll Data Set Results

Landsat Satellite Data Set This database constains multi-spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification associated with the central pixel in the neighborhood. The goal is to predict this classification, using the multi-spectral values. The data set consists of 6435 nodes, each representing a neighborhood of a 82×100 satellite image.

To calculate the weight matrix, we used the weight function in [90] with 30 nearest neighbors. To calculate the eigenvectors, we used the procedure in [1]. The parameters used were: 200 eigenvectors, $C = 51$, $\epsilon = 1$, $dt = 0.1$, $\mu = 50$ and $\eta = 10^{-7}$ with 350 fidelity points (5.44%).

Averaged over 30 runs, the average accuracies obtained were 87.05% and 87.25% for the multiclass GL and MBO methods, respectively. We compare these results to several supervised learning methods listed in [86]. These algorithms were performed using 80% training and 20% validation. The results were: 65.15% using SC-SVM, 75.43% using SH-SVM, 65.88% using S-LS, 86.65% using simplex boosting, and 90.15% using S-LS rbf. We outperform all but the last algorithm using only 5% fidelity, while these algorithms use 80% for training.

Human Activity Data Set This data set from the UCI Machine Learning Repository contains information about experiments carried out with volunteers. Each person performed one of six actions: walking, walking upstairs, walking downstairs, sitting, standing and laying while wearing a smartphone on the waist. The smartphone recorded their linear acceleration and 3-axial angular velocity. The goal is to segment the people into 6 classes according to activity using the information obtained from the phone. The data set has 10229 nodes. We used the weight function in [90] with 59 nearest neighbors. The parameters used were: 50 eigenvectors, $C = 160$, $dt = 0.31$, $\mu = 159$ and $\eta = 10^{-7}$ with 5% of points being fidelity points.

The average accuracy was 89.7% for the multiclass MBO method, while being 88.7% for the GL method. We compare this to the results of [2], where the methods MC-SVM and MC-HF-SVM have accuracies of 89.3% and 89.0%, respectively. It is important to note that the results of the paper were obtained using supervised learning methods where 70% of the data was used for training and the rest for testing. We obtain higher accuracy (with multiclass MBO) using only 5% fidelity.

3.6 Application to Hyperspectral Imagery

We consider the challenge of detection of chemical plumes in hyperspectral image data. Segmentation of gas is difficult due to the diffusive nature of the cloud. The use of hyperspectral imagery provides non-visual data for this problem, allowing for the utilization of a richer array of sensing information. We now present a method to track and classify objects in hyperspectral videos. The method involves the application of a new algorithm recently developed for high dimensional data. It is made efficient by the application of spectral methods and the NystrÂ¨om extension to calculate the eigenvalues/eigenvectors of the graph Laplacian. Results are shown on plume detection in LWIR standoff detection.

Detecting chemical plumes in the atmosphere is a problem that can be applied to many areas, such as defense, security and environmental protection. If the airborne toxins are identified accurately, one can combat the use of chemical gases as weapons, prevent fatalities due to accidental leakage of toxic gases and avoid contamination of the atmosphere. Identification of harmful gases with high fidelity is needed to provide warnings in threatening situations. In these grave scenarios, it is crucial to accurately track the diffusion of dangerous plumes into the atmosphere. Laboratory measured signatures of toxic chemicals are available to assist in chemical plume identification. However, testing and training data is not readily available due to the inherent danger of these real world situations. Instead, open air testing with surrogate chemicals is conducted to study the diffusion of chemical plumes. The developed plume detection methods must meet strict requirements to ensure the fidelity of a detector.

We propose applying the method outlined in [48] to hyperspectral data, in particular, to track and classify chemical plumes, recorded in a hyperspectral video sequence. The pixels of the images in the video are considered as vertices in a graph, and we minimize the total variation with fidelity to known data. The Nyström extension method is used to efficiently calculate eigenfunctions of the graph Laplacian. They are then used both for operator assisted assignment of fidelity values and in the actual total variation minimization algorithm itself. The paper is organized as follows: section 2 reviews the graph representation of the data as well as the Nyström extension method, section 3 presents the method and the results, and the section 4 contains the conclusions.

We consider the data set, described in more detail in [16], composed of video sequences recording the release of chemical plumes at the Dugway Proving Ground. The data was provided by the Applied Physics Laboratory at Johns Hopkins University. The images are of dimension 128 $\times$ 320 $\times$ 129, where the last dimension indicates the number of channels, each depicting a particular frequency from 7,820 nm to 11,700 nm, spaced 30 nm apart. The sets of images were taken from videos captured by three long wave infrared (LWIR) spectrometers, each placed at a different location about 2 km away from the release of plume at an elevation of around 1300 feet. One hyperspectral image is captured every five seconds. This data set has been studied in other works such as [49], [102], [99]. Prior work on hyper spectral plume detection using other sensors includes [65] (MWIR) and [79] (HYDICE). This paper is the first example of the new graphical MBO scheme applied to standoff detection data. The results are excellent compared to prior work in this area.

There are many challenges to be faced when tracking chemical plumes. One obstacle faced by the authors of [49] is the significant preprocessing needed to accurately detect the plume. Due to the noisy structure of the data set, principal component analysis reduced the data to three main features used to produce a false color video sequence of the plume release, followed by midway equalization to smooth the flicker between frames. Similar preprocessing is used in [102], which

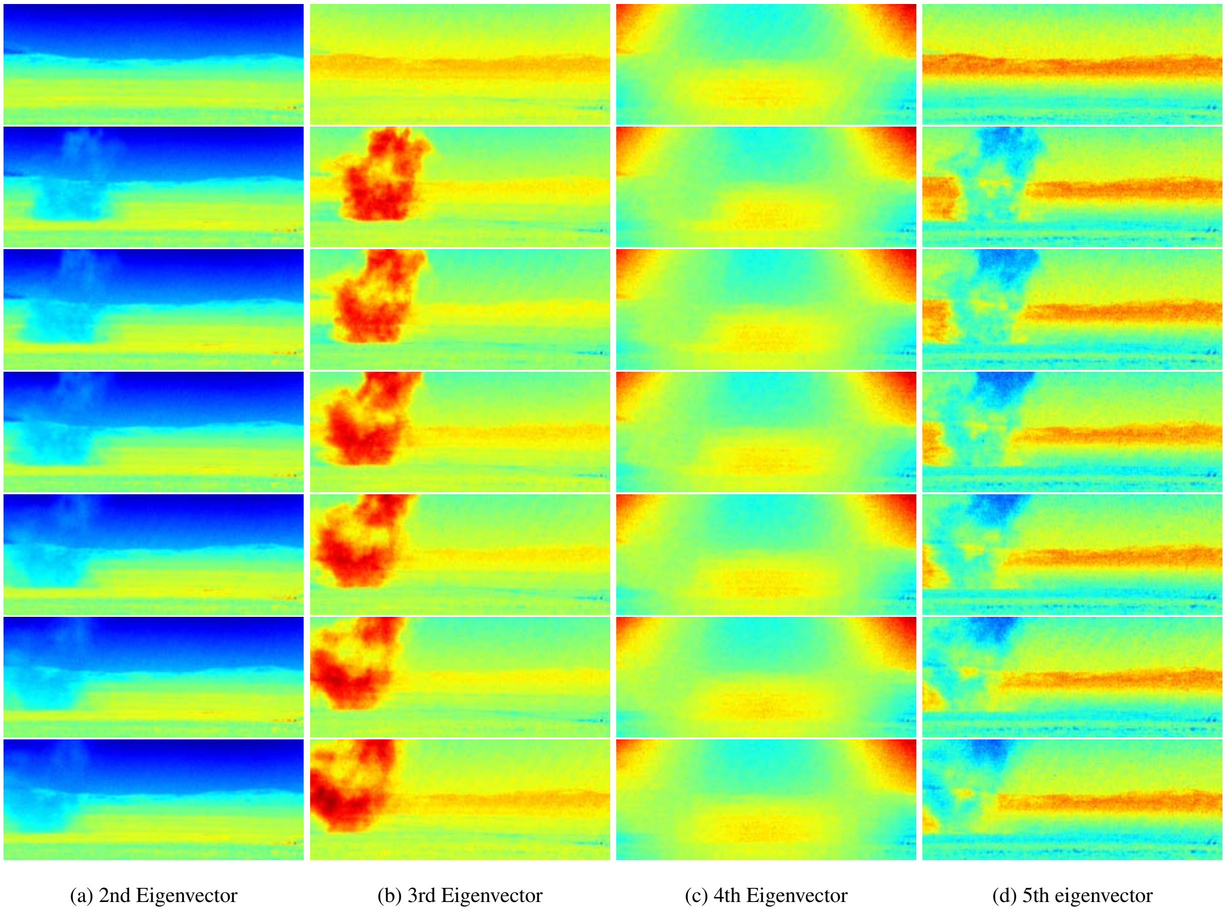
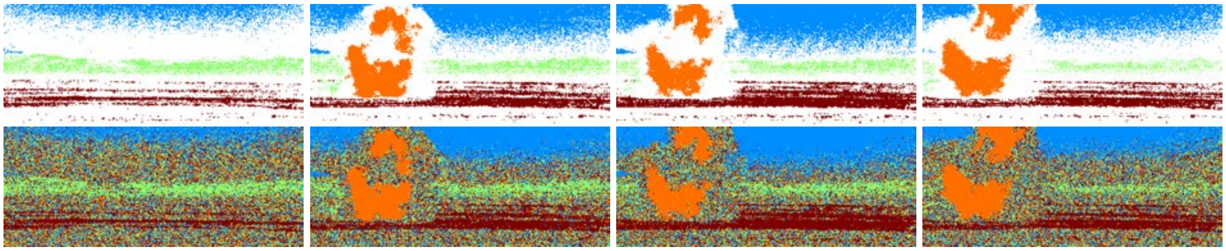

(a) 2nd Eigenvector
(b) 3rd Eigenvector
(c) 4th Eigenvector
(d) 5th eigenvector

Figure 19: Eigenvectors of the symmetric normalized graph Laplacian for the 7 video frames shown in false color.

outlines a novel plume detection method involving a binary partition tree. The advantage of our method is that it does not require any preprocessing of the hyperspectral data; we use the raw data organized in a graph setting. Moreover, as pointed out in [102], the ground truth data is nonexistent, since surrogate chemicals, instead of the toxic ones, are used in testing. This makes the assessment of the results somewhat difficult. The authors of [49] dealt with this problem by using an adaptive matched subspace detector (AMSD) described in [79] to benchmark their spectral clustering results. AMSD is a probabilistic detection scheme that uses a generalized likelihood ratio test to choose between two hypotheses: target present or absent.

We represent the data as nodes in a weighted graph. Since we wish to form a graph that utilizes information inherent in the data over time, so a method of utilizing data from multiple time steps was implemented. This method of performing multiframe analysis is done by selecting k different video frames and then concatenating the data points, allowing for data to be associated over these



(a) Background Frame (b) Gas Frame 1 (c) Gas Frame 2 (d) Gas Frame 3

Figure 20: Fidelity Region and Initialization for the Hyperspectral Video Sequence. First 4 frames of the hyper spectral video: (top row) operator assisted 'ground truth' results from spectral clustering - used as fidelity points in the MBO clustering algorithm; (bottom row) initialization for the MBO scheme. Classification is denoted by color: green = mountain; blue = sky; brown = foreground; orange = plume. White pixels denote unclassified pixels.

k frames. The computation of the graph Laplacian for all of these pixels results in a very large matrix. As an example, the Dugway Proving Ground hyperspectral data is of size 128 $\times$ 320 pixels with 129 spectral bands. For 7 frames, the full graph Laplacian is an $N \times N$ matrix with $N = 286,720$. Thus, a method for quickly computing eigenfunctions of the graph Laplacian is desired. Utilizing the Nyström method, we are able to quickly compute an accurate approximation to the eigenfunctions.

Figure 19 shows a sampling of four different eigenvectors. Note that each eigenvector highlights a different aspect of the image; for example, the third eigenvector outlines the plume. In addition, the background is maintained through the seven different frames. The total run-time for the Nyström extension with 100 eigenvectors is less than one minute on a 2.8 GHz Intel Core 2 Duo. Below we use these eigenvectors for two parts of our multi-class clustering algorithm. In [48], an algorithm is developed to efficiently solve the multiclass assigment problem for graph-based data sets. It is semi-supervised and thus needs "ground truth" assigments for part of the data set. Since we lack real ground truth, we perform operator assisted spectral clustering to obtain partial ground truth. This is performed by identifying the relevant eigenfunctions and thresholding at some level to identify a subset of pixels that are highly likely to be part of the chosen class. For this segmentation we choose four classes: plume, sky, foreground, and mountain.

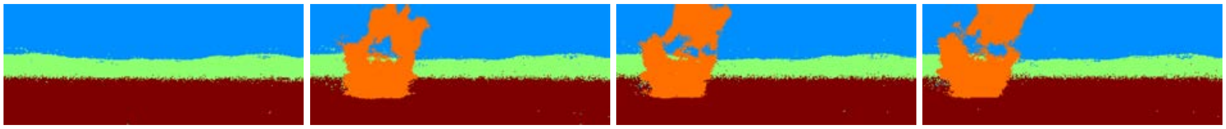

(a) Background Frame (b) Gas Frame 1 (c) Gas Frame 2 (d) Gas Frame 3

Figure 21: Results for the Hyperspectral Video Sequence. First 4 frames, Results of the MBO classification algorithm. Color as specified in figure 20.

Algorithm

We denote the class label vector as the matrix $\mathbf{u}$. The node i adopts a composition of states $\mathbf{u}_i \in \mathbb{R}^K$. Here $K = 4$ for four class segmentation. The k^{th} component of $\mathbf{u}_i$ is the probability the node belongs to class k. For each node i, we require the vector $\mathbf{u}_i$ to be an element of the Gibbs simplex Σ^K, defined as

$$\Sigma^K := \left\{ (x_1, \ldots, x_K) \in [0,1]^K \,\middle|\, \sum_{k=1}^{K} x_k = 1 \right\}. \tag{18}$$

Vertex k of the simplex is given by the unit vector $\mathbf{e}_k$.

It is shown in [48] that alternating between the following two steps results in an efficient classification algorithm:

1. Heat equation with forcing term:

$$\frac{\mathbf{u}^{n+\frac{1}{2}} - \mathbf{u}^n}{dt} = -\mathbf{L}_s \mathbf{u}^{n+\frac{1}{2}} - \boldsymbol{\mu}(\mathbf{u}^n - \hat{\mathbf{u}}). \tag{19}$$

2. Thresholding:

$$\mathbf{u}_i^{n+1} = \mathbf{e}_k,$$

   where $\mathbf{e}_k$ is the vertex in the simplex closest to the projection of $\mathbf{u}_i^{n+\frac{1}{2}}$ onto the simplex using [23].

Here μ_i is a positive constant μ if node i's label is known beforehand (fidelity point) and 0 otherwise, and $\hat{\mathbf{u}}_i$ is a vector indicating prior class knowledge of sample i. Also, note that in the second step the row vector $\mathbf{u}_i{}^{n+\frac{1}{2}}$ of step 1 is projected back to the simplex before any thresholding takes place. This is done because the result of step 1 is not necessary an element of the Gibbs simplex.

Scheme (19) is solved using the eigenvalue/eigenvector decomposition of the symmetric graph Laplacian. The Laplacian term is treated implicitly. The first part of the algorithm can be rewritten as

$$\mathbf{u}^{n+\frac{1}{2}} = (\mathbf{I} + dt\,\mathbf{L}_{\boldsymbol{s}})^{-1}(\mathbf{u}^n - dt\boldsymbol{\mu}(\mathbf{u}^n - \hat{\mathbf{u}})). \tag{20}$$

We use the eigendecomposition $\mathbf{L}_{\boldsymbol{s}} = \mathbf{X}\boldsymbol{\Lambda}\mathbf{X}^T$ to write

$$\mathbf{I} + dt\,\mathbf{L}_{\boldsymbol{s}} = \mathbf{X}\,(\mathbf{I} + dt\,\boldsymbol{\Lambda})\,\mathbf{X}^T, \tag{21}$$

but we approximate $\mathbf{X}$ by a truncated matrix retaining only N_e eigenvectors ($N_e \ll N_D$), to form a matrix of dimension $N_D \times N_e$. The term in the parenthesis in (21) is a diagonal $N_e \times N_e$ matrix. This allows one to calculate $\mathbf{u}^{n+\frac{1}{2}}$ rapidly. In particular, we write, for the n^{th} iteration, $\mathbf{u}^n = \mathbf{X}\mathbf{a}^n$ and $\boldsymbol{\mu}(\mathbf{u}^n - \hat{\mathbf{u}})) = \mathbf{X}\mathbf{d}^n$, where $\mathbf{a}$ and $\mathbf{d}$ are matrices of dimension N_e by K, where K is the number of classes. Denote $\mathbf{E}$ to be the diagonal matrix containing the eigenvalues of the symmetric graph Laplacian, then $\mathbf{L_s}\mathbf{u}^n = \mathbf{X}\mathbf{E}\mathbf{a}^n$. Also denote by $\mathbf{a}_k$ and $\mathbf{d}_k$ the k^{th} row of $\mathbf{a}$ and $\mathbf{d}$, respectively. Plugging all the known expressions into (19), we obtain an equation for $\mathbf{a}_k^{n+1}$ that effectively replaces (19):

$$\boxed{\mathbf{a}_k^{n+1} = \frac{\mathbf{a}_k^n - dt\mathbf{d}_k^n}{1 + dt\lambda_k},}$$

where λ_k is the k^{th} eigenvalue of the symmetric graph Laplacian. The remaining step is simple thresholding.

We tested our method on seven video frames, using four classes, to segment the plume, sky, foreground and mountain. The fidelity region is shown in the first row of figure 20. The initialization for the MBO scheme is displayed in the second row. The final segmentation results, after 17 iterations, are shown in figure 21, using first four frames.

The fidelity region is calculated differently from [48], where the fidelity points were chosen randomly from known ground truth data. Without the ground truth, we use an operator assisted method involving spectral clustering. In particular, by thresholding appropriately the values of eigenvectors, one obtains information about a particular class. For example, as shown in Figure 19, the third eigenvector provides information about the plume. By thresholding its values, one can find the pixels that are most likely part of the plume. We used the fifth eigenvector to obtain fidelity for the mountain, and the second one for both the sky and the foreground. This process resulted in 36% points of the overall points from all the frames identified as as good fidelity points.

For the MBO scheme, we start with an initialization of randomly chosen phase classes for non-fidelity points and the "ground truth" value for the fidelity points. The MBO iteration was performed 17 times using a stopping criterion for convergence, with $dt = 0.1$ and $\boldsymbol{\mu} = 100$. We compared results with 10 to 100 eigenvectors; they deteriorated with less than 10 eigenvectors but were similar with more than 10 eigenvectors. In all tests we used the same operator assisted fidelity points. The MBO iteration took around 11 seconds (for 100 eigenvectors) on a 2.4 GHz Intel Core i2 Quad, after obtaining the eigenvectors from the Nyström scheme.

We thus presented an application of a recent multiclass classification algorithm [48] to hyperspectral video data. We use the Nyström extension method to efficiently calculate the needed eigenvectors. This implementation of the algorithm requires an operator assisted spectral clustering preprocessing step to identify a subset of pixels denoted as "ground truth" for the four classes. The resulting classification of chemical plumes and background pixels are excellent. Only a small number of eigenvectors, ten in particular, is needed to achieve a good result and no preprocessing is necessary. The entire process took about a minute on desktop PCs.

Table 3: Multiclass MBO and GL Method Classification Results

Two/Three moons

Method	Accuracy
spectral clustering [47]	80%
p-Laplacian [18]	94%
Cheeger cuts [100]	95.4%
tree GL [47]	97.4%
binary GL [9]	97.7%
multiclass GL	98.4%
multiclass MBO	99.12%

MNIST

Method	Accuracy
p-Laplacian [18]	87.1%
multicut normalized 1-cut [64]	87.64%
linear classifiers [71, 72]	88%
Cheeger cuts [100]	88.2%
boosted stumps [68, 72]	92.3-98.74%
transductive classification [101]	92.6%
tree GL [47]	93.0%
k-nearest neighbors [71, 72]	95.0-97.17%
neural/convolutional nets [25, 71, 72]	95.3-99.65%
nonlinear classifiers [71, 72]	96.4-96.7%
multiclass GL	96.8%
multiclass MBO	96.91%
SVM [29, 71]	98.6-99.32%

COIL

Method	Accuracy
k-nearest neighbors [98]	83.5%
LapRLS [6, 98]	87.8%
sGT [67, 98]	89.9%
SQ-Loss-I [98]	90.9%
MP [98]	91.1%
multiclass GL	91.4%
multiclass MBO	91.46%

WebKB

Method	Accuracy
vector method [20]	64.47%
k-nearest neighbors ($k = 10$) [20]	72.56%
centroid (normalized sum) [20]	82.66%
naive Bayes [20]	83.52%
SVM (linear kernel) [20]	85.82%
multiclass GL	87.3%
multiclass MBO	88.48%

Other Data Sets

Method	Accuracy multiclass MBO	Accuracy multiclass GL
swissroll data set	91.8%	94.1%
Landsat satellite data set	87.25%	87.05%
human activity data set	89.7%	88.7%

Table 4: MBO Method WebKB Results

Method	10%	15%	20%	25%	30%
WebKB results for Multiclass GL (% correct)	81.5%	84.2%	85.4%	86.7%	87.3%
WebKB results for Multiclass MBO (% correct)	83.71%	85.75%	86.81%	87.74%	88.48%

Table 5: Runtime (in Seconds) of the MBO Method

Data set	three moons	grayscale image	color image	MNIST	COIL	WebKB
Graph Calculation	0.771	19.96	645.34	6183.1	0.95	399.35
Eigenvector Calculation	0.331	210.10	190.93	1683.5	0.19	64.78
Multiclass GL	0.163	4.08	19.92	811.5	2.31	6.97
Multiclass MBO	0.013	0.23	1.20	15.4	0.03	0.05

Table 6: Number of Iterations of the MBO Method

Data set	three moons	grayscale image	color image	MNIST	COIL	WebKB
Multiclass GL	140	90	200	460	700	275
Multiclass MBO	3	2	11	7	6	7

Table 7: Confusion Matrix for the MNIST Data Set- Multiclass GL

Obtained/True	0	1	2	3	4	5	6	7	8	9
0	6844	1	43	4	3	16	21	2	19	19
1	6	7809	36	8	35	2	14	62	58	15
2	5	22	6733	45	2	4	1	27	19	7
3	0	3	20	6882	1	91	0	1	89	92
4	1	16	6	2	6626	4	7	15	28	75
5	9	0	3	75	0	6072	28	2	125	14
6	31	5	11	3	23	65	6802	0	31	4
7	2	16	108	45	11	7	0	7078	19	110
8	1	2	22	42	4	15	3	2	6365	20
9	4	3	8	35	119	37	0	104	72	6602

Table 8: Confusion Matrix for the MNIST Data Set- Multiclass MBO

Obtained/True	0	1	2	3	4	5	6	7	8	9
0	6844	20	41	3	3	15	21	1	20	17
1	5	7789	32	8	34	1	14	63	51	14
2	5	22	6731	42	2	4	1	23	19	8
3	0	3	20	6890	1	86	0	1	81	90
4	1	17	6	2	6625	3	7	12	28	67
5	9	0	3	70	0	6077	28	2	109	14
6	31	5	11	3	22	69	6800	0	29	5
7	2	16	117	44	12	9	0	7093	20	101
8	2	2	21	46	4	17	5	2	6398	22
9	4	3	8	33	121	32	0	96	70	6620

CHAPTER 4

Convex Method

In this section, several versions of a convex binary classification method are developed. We are interested in solving partition problems of the form

$$\min_{S \subset V} \sum_{(x,y)\in E \,:\, x\in S,\; y\in V\backslash S} w(x,y), \tag{22}$$

which is the formulation of the minimum cut problem, under supervised constraints

$$S \supseteq V^f, \quad V\backslash S \supseteq V^b \tag{23}$$

and an optional volume constraint

$$|S| = a|V|, \tag{24}$$

where $0 < a < 1$. $V^f \subset V$ is a set of nodes that are known a priori to belong to the region S and $V^b \subset V$ is a set of nodes that are known to belong to region $V\backslash S$. Variations of this problem have been vastly explored in literature. For example, in [14], the authors describe an algorithm which minimizes a normalized version of the cut, specifically the balanced cut. A multiclass version of this method is introduced in [13].

By defining a binary function

$$u(x) \;=\; \begin{cases} 1, & x \in S \\ 0, & x \in V\backslash S, \end{cases}$$

the supervised minimum cut problem can be regarded as

$$\min_{u\in\mathcal{B}} E^P(u) = TV_w(u) + \sum_{x\in V} f(u(x),x), \tag{25}$$

where

$$TV_w(u) = \frac{1}{2}\sum_{x.y} w(x,y)|u(x)-u(y)|$$

as defined earlier (with $q = 1$) and

$$\mathcal{B} = \{u : V \mapsto \{0,1\}\} \tag{26}$$

is the set of binary functions indicating the partition. Here, $f(u(x), x)$ is a fidelity term which incorporates the supervised constraints (23). It typically takes the form of

$$f(u(x), x) = \eta(x)|u(x) - u^0(x)|^2, \tag{27}$$

where u^0 is a binary function taking value 1 in V^f and 0 in V^b, and $\eta(x)$ is a function that takes on a large constant value η on fidelity points $V^f \cup V^b$ and zero elsewhere. If η is chosen sufficiently large, it can be guaranteed that the solution u satisfies the supervised constraints. In [75], the authors propose solving a similar minimization problem by introducing nonlocal global minimizers of active contour models on graphs.

In addition, when the size of the two classes is known, the volume of the regions may be enforced to satisfy a constraint of the form

$$\sum_{x \in V} u(x) = a|V|, \tag{28}$$

where a is the fraction of the nodes belonging to class 2, e.g. $a = \frac{1}{2}$ enforces partitions of equal volume. The goal of this is to create an algorithm that requires a much smaller fidelity set (to produce an accurate classification) than otherwise, because we also have the information about class size.

In previous work [9], the problem (25) was formulated as the minimization of a Ginzburg-Landau (GL) functional on graphs with a fidelity term

$$GL_\epsilon(u) = \|\nabla u\|_{\mathcal{E}}^2 + \frac{1}{\epsilon} \sum_x W(u(x)) + f(u(x), x), \tag{29}$$

where

$$\|\nabla u\|_{\mathcal{E}}^2 = \frac{1}{2} \sum_{x,y} w(x, y)(u(x) - u(y))^2$$

as defined before. Note that as $\epsilon \to 0$, in the limit of gamma convergence, the sum of first two terms of the energy converge to the total variation term, making the energy exactly the same as one in (25). The problem is solved using gradient descent and an efficient convex splitting scheme. This method will be referred to as "binary GL" in the paper, and we compare it to our work.

In [80], (29) is solved numerically by a variation of the MBO scheme [81], a method to approximate motion by mean curvature. To make everything consistent with the notation and theorems stated in the paper, we include an extra scaling in our implementation of the method in [80], and the justification is described shortly. We note that this change in the method did not exacerbate the results as compared to those of the original method; in fact, it produced very little change in any simulation. This algorithm will be referred to as "binary MBO" in the paper, and we compare it to our new algorithms. The discretized version of the algorithm is the following:

Starting with some initial classification $u \in \{0, 1\}$, alternate between the following two steps until the stopping criterion is satisfied:

1. Heat equation with forcing term:

$$\frac{u^{n+\frac{1}{2}} - u^n}{dt} = 2\triangle_w u^{n+1} - \frac{1}{d(x)^r}\frac{\partial f(u(x), x)}{\partial u}. \tag{30}$$

2. Thresholding:

$$u^{n+1}(x) = \begin{cases} 1, & \text{if } u^{n+\frac{1}{2}}(x) \geq 0.5, \\ 0, & \text{if } u^{n+\frac{1}{2}}(x) < 0.5. \end{cases}$$

Here, after the second step, $u^{n+1}(x)$ can take only two values of 1 or 0; thus, this method is appropriate for binary segmentation.

Following [80], (30) is solved by a semi-implicit scheme, where the Laplacian term is calculated implicitly, and the terms are considered as a linear combination of the eigenvectors of the random walk Laplacian.

To show the general idea of the derivation, we start with the Ginzburg-Landau (GL) functional on graphs (29). One can rewrite it using inner product notation:

$$GL_\epsilon(u) = \|\nabla u\|^2_{\mathcal{E}} + \frac{1}{\epsilon}\langle D^{-r}W(u(x)), 1\rangle_{\mathcal{V}} + \langle D^{-r}f(u(x), x), 1\rangle_{\mathcal{V}},$$

where $(D^{-r}W(u))(x) = d(x)^{-r}W(u(x))$. The factor $d(x)^{-r}$ is needed to cancel the factor $d(x)^r$ in the $\mathcal{V}$- inner product.

The modified Allen-Cahn equation can then be derived using the $\mathcal{V}$-gradient flow associated with GL_ϵ. We have

$$\frac{d}{dt}GL_\epsilon(u + t\gamma)|_{t=0} = -2\langle\triangle_w u, \gamma\rangle_{\mathcal{V}} + \frac{1}{\epsilon}\langle D^{-r}W'(u(x)), \gamma\rangle_{\mathcal{V}} + \langle D^{-r}\frac{\partial f}{\partial u}(u(x), x), \gamma\rangle_{\mathcal{V}}.$$

The modified Allen-Cahn equation is then

$$\dot{u}(x) = 2\triangle_w u - \frac{1}{\epsilon d(x)^r}W'(u(x)) - \frac{1}{d(x)^r}\frac{\partial f}{\partial u}(u(x), x).$$

The above equation can be solved using a time-splitting scheme, where the splitting occurs so that the double-well potential term is separated. The first step is

$$\dot{u}(x) = 2\triangle_w u - \frac{1}{d(x)^r}\frac{\partial f}{\partial u}(u(x), x)$$

and the second step (with the double-well potential) is just thresholding in the $\epsilon \to 0$ limit. By alternating between these two steps, one can form an approximate solution of the modified Allen-Cahn equation (4).

Such approaches (binary GL and binary MBO methods) converge to the nearest local minimizer from a given initialization. In general, one cannot guarantee that the desired global minimizer is obtained. The subject of this work is to develop a convex optimization framework for minimizing (25) which is guaranteed to obtain the global minimizer. Various algorithms are developed for solving the convex problems.

The problem (25) is non-convex because the binary side constraints (26) are non-convex. We show that the binary constraints can be replaced by their convex hull $[0, 1]$ to obtain an exact convex formulation as was shown in [21] for images. Define first the functions

$$C_s(x) = f(0, x)\,, \quad C_t(x) = f(1, x), \quad \forall x \in V,$$
$$g(\phi(x), x) = C_t(x)\phi(x) + C_s(x)(1 - \phi(x))\,,\ \forall x \in V. \tag{31}$$

The problem

$$\min_{u \in \mathcal{B}} E(u) = TV_w(u) + \sum_{x \in V} g(u(x), x) \tag{32}$$

is equivalent to the formulation (25). The proof is obvious as $g(\phi(x), x) = f(\phi(x), x)$ for all binary ϕ.

The convex relaxed problem is formulated as follows:

$$\boxed{\min_{u \in \mathcal{B}'} E(u) = TV_w(u) + \sum_{x \in V} g(u(x), x),} \tag{33}$$

where

$$\mathcal{B}' = \{u \,:\, V \mapsto [0, 1]\}. \tag{34}$$

This is the problem we will be concerned with in this section about the convex method.

The following theorem, which is a generalization of theorem 1 in [21] from images to graphs, shows that the minimizer of the convex problem can be thresholded to yield a binary global minimizer of the original problem.

Theorem 2. *Let u^* be a minimizer of* (33). *Denote by $u^\ell \,:\, V \mapsto \{0, 1\}$ the binary function*

$$u^\ell(x) = \begin{cases} 1\,, & \textit{if}\, u^*(x) \geq \ell. \\ 0\,, & \textit{if}\, u^*(x) < \ell. \end{cases}\,.$$

Then for almost every $\ell \in (0,1]$, u^ℓ is a global minimizer of the non-convex problem (32).

Proof: For any function $u \in \mathcal{B}'$ and for any $x \in V$, if $u(x) \in [0,1]$, then $\int_0^1 u^\ell(x)\,d\ell = u(x)$. Therefore, for each $x \in V$,

$$\int_0^1 g(u^\ell(x),x)\,d\ell = \int_0^1 u^\ell(x)C_t(x)+(1-u^\ell(x))C_s(x)\,d\ell = u(x)C_t(x)+(1-u(x))C_s(x) = g(u(x),x).$$

By the coarea formula, we have that

$$\int_0^1 TV_w(u^\ell)\ d\ell = TV_w(u)\,.$$

This can be easily shown using the definition of TV_w. In particular, we see that

$$\int_0^1 TV_w(u^\ell)\ d\ell = \int_0^1 \frac{1}{2}\sum_{x,y} w(x,y)|u^\ell(x)-u^\ell(y)|dx =$$
$$\frac{1}{2}\sum_{x,y} w(x,y)\int_0^1 |u^\ell(x)-u^\ell(y)|dx = \frac{1}{2}\sum_{x,y} w(x,y)|u(x)-u(y)| = TV_w(u).$$

The above can be seen using the fact that (assuming that $u(x) < u(y)$ without loss of generality)

$$|u^\ell(x)-u^\ell(y)| = \begin{cases} 0 & \text{if } \ell \le x. \\ 1 & \text{if } u(x) < \ell < u(y). \\ 0 & \text{if } \ell \ge y. \end{cases}$$

so that $\int_0^1 |u^\ell(x)-u^\ell(y)|dx = |u(x)-u(y)|$.

For a proof of the coarea formula in the continuous case, see Appendix B of [104].

Combining the above properties, we obtain that for any $u \in \mathcal{B}'$,

$$\int_0^1 E^P(u^\ell)\,d\ell = \sum_{x\in V}\int_0^1 (TV_w(u^\ell)+g(u^\ell(x),x))d\ell = \sum_{x\in V} TV_w(u)+g(u(x),x) = E^P(u).$$

For a u that minimizes the energy, clearly $E^P(u) \le E^P(u^\ell)$ for any $\ell \in (0,1]$. However, equality (4) can then only be true provided $E^P(u) = E^P(u^\ell)$ for almost every $\ell \in (0,1]$. In other words, u^ℓ also minimizes the energy for almost every $\ell \in (0,1]$. ■

In order to solve (33), we consider two main algorithms. The first is based on solving an equivalent formulation of the problem, which can be identified as a maximum-flow problem, by convex optimization techniques. It will be referred to as the "max-flow" method in this paper. We present three versions of this algorithm: one without hard supervised constraints (Convex Method: Version 1), one with hard supervised constraints (Convex Method: Version 1s), and one with balancing constraints (Convex Method: Version 1b). The second algorithm (Convex Method: Version 2) solves the original problem by the augmented Lagrangian technique, and will be denoted the “primal augmented Lagrangian” method in this section.

4.1 Convex Method (Versions 1 and 1s): Max-flow *Without* Balancing Constraints

In this subsection, we introduce the first version of solving (33), which consists of formulating the problem in an equivalent max-flow setting using a graphical framework.

We have discussed earlier the general graph framework where one considers a set of vertices V and the set of edges E between them. We now extend it further by adding two “special” vertices, the source and the sink, to which the regular vertices are connected. Let us now view this problem in a more physical setting by considering a flow that passes from the source to the sink through the vertices in between. Let the weight defined on each edge represent the upper bound on the amount of flow that is allowed to pass through the edge. Of course, that capacity does not need to be reached, this is just the upper bound.

The max-flow problem [43] aims to maximize the flow coming from the source node s to a sink node t. We let $p_s, p_t : V \mapsto \mathbb{R}$ denote the amount of flow on the edges that connect a source or sink edge, respectively, to a regular vertex of the graph. For example, $p_s(x)$ denotes how much flow is passing from the source to vertex x. Analogously, $p_t(x)$ denotes how much flow is passing from vertex x to the sink. Now let $p : E \mapsto \mathbb{R}$ represent the amount of flow on edges between pairwise points in V, where $E \subset V \times V$. For example, $p(x, y)$ denotes the amount of flow passing from vertex x to vertex y. The upper capacities on the source edges are denoted by C_s and on sink edges by C_t, and there is no lower bound on the capacities. Therefore,

$$p_s(x) \leq C_s(x) \quad \forall x \tag{35}$$

and

$$p_t(x) \leq C_s(x) \quad \forall x. \tag{36}$$

The flows $p(x, y)$ on the edges (x, y) are bounded by $|p(x, y)| \leq C$. In this section, we choose C to be 1. However, one does not have to use a constant bound here; one can consider a general function $C(x, y)$. The amount of flow in the graph can be expressed as the amount of flow on the source edges, which we want to maximize under flow capacity and flow conservation constraints.

In this section, we describe two max-flow problems. The first is equivalent to the problem (25) with fidelity term, and consequently solves the original problem (22) provided the penality parameter η is high enough. The second max-flow problem incorporates the supervised constraints directly without the need for a very large penalty term. The following derivations extend the continuous max-flow problem [108, 109] from images to general graphs. All the operators are written in graph form.

4.1.1 Max-flow Formulation with Supervised Constraints as Fidelity Term

In this subsection, we show that the following *primal* formulation (that can be interpreted as a max-flow problem over the graph) is equivalent to the convex partition problem (33):

$$\max_{p_s, p_t, p} \left\{ P(p_s, p_t, p) \, = \, \sum_{x \in V} p_s(x) \right\} \tag{37}$$

subject to

$$|p(x,y)| \, \leq \, C, \quad (x,y) \in E; \tag{38}$$

$$p_s(x) \, \leq \, C_s(x), \quad \forall x \in V; \tag{39}$$

$$p_t(x) \, \leq \, C_t(x), \quad \forall x \in V; \tag{40}$$

$$\mathrm{div}_w \, p(x) - p_s(x) + p_t(x) \, = \, 0, \quad \forall \, x \in V. \tag{41}$$

where (38) is the flow capacity constraint on edges between paiwise nodes, (39) and (40) are flow capacities on the source and sink edges, and (41) is the flow conservation constraint. Note that we use the divergence operator to evaluate the net flow locally around vertex x. The method of solving the formulation is outlined as “Convex Method: Version 1”, but will also be referred to as the max-flow algorithm.

The objective function (37) measures the total amount of flow on the graph. Due to constraint (39), the maximization problem (37) is bounded above by $\int_\Omega C_s(x)$, which is finite provided $f(\phi(x), x)$ is bounded (true for the data terms considered in this work).

It is well known that the value of the maximum flow of the maximum flow problem is equivalent to the value of the minimum cut (where one sums the weights on the edges between vertices of different classes). In classical max-flow min-cut theory, to obtain the final classification by solving the maximum-flow problem, one can use the information of the flow on the source and sink edges.

If for $x \in V$, there is a non-saturated path between s and x, then x is in class 1. If there is a non-saturated path between x and t, then x is in class 2.

In this paper, we instead solve the max-flow problem (37) by continuous optimization. Introducing such a Lagrange multiplier (see background section on optimizing function) for the flow conservation constraint (41) leads to the the $primal - dual$ formulation of (37):

$$\min_{\lambda} \max_{p_s, p_t, p} \Big\{ E(p_s, p_t, p; \lambda) = \sum_{x \in V} p_s(x) + \sum_{x \in V} \lambda(x) \big(\operatorname{div}_w p - p_s + p_t \big) \Big\} \tag{42}$$

subject to

$$|p(x,y)| \leq C, \quad (x,y) \in E; \tag{43}$$

$$p_s(x) \leq C_s(x), \quad \forall x \in V; \tag{44}$$

$$p_t(x) \leq C_t(x), \quad \forall x \in V. \tag{45}$$

Rearranging the terms, we obtain

$$\min_{\lambda} \max_{p_s, p_t, p} \Big\{ E(p_s, p_t, p; \lambda) \; = \; \sum_{x \in V} \big\{ \big(1 - \lambda\big) p_s + \lambda p_t + \lambda \operatorname{div}_w p \big\} \Big\} \tag{46}$$

subject to (43), (44) and (45).

By maximizing the above problem for p_s, p_t and p, in the continuous (or in the graph) case, we obtain the closed form expression (33) subject to the constraint (34). If the constraint (34) was not met, the energy could become arbitrarily large by chosing p_s or p_t arbitrarily low, contradicting boundedness from above.

In order to optimize the problem (46), let us first consider a continuous setting and the problem

$$f(q) = \max_{p \leq C} p \cdot q. \tag{47}$$

When $q < 0$, we can choose p to be negative infinity to maximize $p \cdot q$, which gives $f(q) = +\infty$. Also, if $q = 0$, then $p \leq C$ and $f(q)$ reaches maximum at 0. If $q > 0$, then $p = C$ and $f(q)$ reaches maximum at $q \cdot C$. Therefore,

$$f(q) = \begin{cases} \infty & \text{if } q < 0. \\ q \cdot C & \text{otherwise.} \end{cases} \tag{48}$$

Define

$$f_s(x) = \max_{p_s(x) \leq C_s(x)} (1 - \lambda(x)) \cdot p_s(x), \tag{49}$$

$$f_t(x) = \max_{p_t(x) \le C_t(x)} \lambda(x) \cdot p_t(x). \tag{50}$$

Then we have the following:

$$f_s(x) = \begin{cases} (1-\lambda(x)) \cdot C_s(x) & \text{if}(1-\lambda(x)) \ge 0, \\ \infty, & \text{otherwise} \end{cases} \tag{51}$$

and

$$f_t(x) = \begin{cases} \lambda(x) \cdot C_s(x) & \text{if}\lambda(x) \ge 0, \\ \infty, & \text{otherwise} \end{cases} \tag{52}$$

By [53], it is known that

$$\max_{p(x)| \le C} \int_\Omega \lambda \operatorname{div} p dx = \int_\Omega K |\nabla \lambda| dx, \tag{53}$$

where K is a constant. Here, we write $p(x)$ because this is a continuous setting, defined on a plane.

We can also derive an equivalent of this equation in a graph setting. We have

$$\langle \lambda, \operatorname{div}_w p \rangle_V = \sum_x \frac{1}{2} \lambda(x) \{ \sum_y w(x,y)(p(x,y) - p(y,x)) \} =$$

$$\sum_{x,y} \lambda(x) \frac{1}{2} w(x,y) p(x,y) - \sum_{x,y} \frac{1}{2} \lambda(x) w(x,y) p(y,x) = \sum_{x,y} \frac{1}{2} (\lambda(x) - \lambda(y)) w(x,y) p(x,y).$$

Since

$$\max_{p(x,y) \le C} \sum_{x,y} \frac{1}{2} (\lambda(x) - \lambda(y)) w(x,y) p(x,y) = \frac{1}{2} \sum_{x.y} w(x,y) |\lambda(x) - \lambda(y)|,$$

we see that

$$\max_{p(x,y) \le C} \langle \lambda, \operatorname{div}_w p \rangle_V = K \cdot TV_w(\lambda), \tag{54}$$

where K is a constant.

By (51),(52) and (54), in a continuous setting, maximization of (46) over flows p_s, p_t and p leads to the equivalent dual model (33) subject to the constraint (34). We thus have the following **equivalent** formulations:

Primal Formulation

$$\max_{p_s, p_t, p} \left\{ P(p_s, p_t, p) \ = \sum_{x \in V} p_s(x) \right\}$$

subject to (38)- (41).

Primal-Dual Formulation

$$\min_\lambda \max_{p_s, p_t, p} \left\{ E(p_s, p_t, p; \lambda) = \sum_{x \in V} p_s(x) + \sum_{x \in V} \lambda(x) \big(\operatorname{div}_w p - p_s + p_t \big) \right\}$$

subject to (38)- (40).

Dual Formulation

$$\min_{\{\lambda : V \mapsto [0,1]\}} E(u) = TV_w(\lambda) + \sum_{x \in V} g(\lambda(x), x).$$

Existence of dual and primal-dual solutions follows by the minimax theorem, Prop. 2.4 of [35] Chapter VI. In more detail, the constraints of the flows are convex, and the function is linear in all the variables, and thus convex l.s.c. for a fixed λ and concave u.s.c. for fixed p_s,p_t and p. Therefore, we have the existence of at least one saddle point. Also, the min and max operators of the primal model can be exchanged:

$$\min_{\lambda} \max_{p_s, p_t, p} \{E(p_s, p_t, p; \lambda) = \max_{p_s, p_t, p} \min_{\lambda} \{E(p_s, p_t, p; \lambda)\}. \tag{55}$$

When we maximize the primal-dual problem over λ, we obtain the primal model:

$$P(p_s, p_t, p) = \min_{\lambda} \{E(p_s, p_t, p; \lambda)\}. \tag{56}$$

Note: Connection between variables

Solving the primal problem, how can one obtain λ or the solution to the min-cut problem? Let $(\lambda^*(x), p_s^*, p_t^*, p)$ be the optimal primal-dual pair of (46). Now observe that if $p_s(x) < C_s(x)$ (flow is unsaturated), we have $1 - \lambda^*(x) = 0$, i.e.

$$p_s^*(x) < C_s(x) \Longrightarrow \lambda^*(x) = 1. \tag{57}$$

Therefore, we class vertex x as being of class 2. Morever, $f_s(x) = (1 - \lambda^*(x))p_s^*(x) = 0$, so that $p_t^*(x) = C_t(x)$. Thus, that flow is saturated.

Similarly, in the case that $p_t^*(x) < C_t(x)$ (flow is unsatured), we have $\lambda^*(x) = 0$, i.e.

$$p_t^*(x) < C_t(x) \Longrightarrow \lambda^*(x) = 0. \tag{58}$$

Therefore, we class vertex x as being of class 1. Morever, $f_t(x) = \lambda^*(x)p_s^*(x) = 0$, so that $p_s^*(x) = C_s(x)$. Thus, the flow from the source is saturated.

One can also obtain obtain the solution to the min-cut problem using the primal-dual problem by the Lagrange multiplier or λ.

Therefore, the remarks above show an elegant relationship between λ and the three variables of the primal problem.

4.1.2 Max-flow Formulation with Hard Supervised Constraints

We also describe another formulation of the problem, which avoids using a fidelity term that is forced to take on a very large value of η to enforce that the supervised constraints are satisfied. Define first the binary functions

$$v^f(x) = \begin{cases} 1, & x \in V^f \\ 0, & \text{otherwise} \end{cases}, \quad v^b(x) = \begin{cases} 0, & x \in V^b \\ 1, & \text{otherwise} \end{cases},$$

and consider the following modification of the max-flow problem (37)

$$\max_{p_s, p_t, p} \left\{ P(p_s, p_t, p) = \sum_{x \in V} (v^b p_s - v^f p_t) \right\} \tag{59}$$

subject to

$$|p(x,y)| \le C, \quad (x,y) \in E; \tag{60}$$

$$p_s(x) \le 0, \quad \forall x \in V; \tag{61}$$

$$p_t(x) \le 0, \quad \forall x \in V; \tag{62}$$

$$\operatorname{div}_w p(x) - p_s(x) + p_t(x) = 0, \quad \forall\, x \in V. \tag{63}$$

Introducing the Lagrange multiplier λ for constraint (63), we obtain the following Lagrangian formulation after rearrangement of the terms:

$$\min_{\lambda} \max_{p_s, p_t, p} \left\{ E(p_s, p_t, p; \lambda) = \sum_{x \in V} \left\{ \left(v^b - \lambda\right) p_s + (\lambda - v^f) p_t + \lambda \operatorname{div}_w p \right\} \right\}. \tag{64}$$

As there are no lower bounds on p_s and p_t, it can be observed that optimal solutions λ must satisfy the constraints

$$v^f \le \lambda \le v^b, \tag{65}$$

otherwise the energy could be made arbitrarily large. Note that, by maximizing for the flows p_s, p_t and p in continuous space, we therefore obtain the primal problem

$$\min_{\lambda \in B'} TV_w(\lambda) \tag{66}$$

subject to (65). See previous section for the ideas of the derivation.

Also, if λ^* is a solution to (66) subject to supervised constraints, then λ^* is obviously also a solution to (33) provided the penalty parameter η in the fidelity term of (33) is chosen sufficiently high.

The method of solving the formulation is outlined as “Convex Method Version 1s”.

Algorithms

The dual problems (37) or (59) are solved by the augmented Lagrangian method as in [108,109]. To solve (37), construct first the augmented Lagrangian function corresponding to (37):

$$L_c(p_s, p_t, p, \lambda) = \sum_{x \in V} \left\{ p_s + \lambda(\operatorname{div}_w p - p_s + p_t) \right\} - \frac{c}{2} \left\| \operatorname{div}_w p - p_s + p_t \right\|_2^2 ,$$

where $\|s\|_2^2 = \sum_{x \in V} |s(x)|_2^2$.

The goal is to solve

$$\max_{p_s, p_t, p} \left\{ P(p_s, p_t, p) \; = \sum_{x \in V} p_s(x) \right\}$$

under constraints (38) - (41) by solving

$$\min_{\lambda} \max_{p_s, p_t, p, \lambda} L_c(p_s, p_t, p, \lambda).$$

An augmented Lagrangian method can be applied to solve this problem by alternatively maximizing L_c for the dual variables p_s, p_t and p with constraints (43)-(45) and updating the Lagrange multiplier λ as follows:

Convex Method (Version 1) Max-flow Algorithm

Initialize p_s^1, p_t^1, p^1 and λ^1. For $k = 1, ...$ until convergence:

- Optimize p flow

$$p^{k+1} = \underset{|p(e)| \le C \ \forall e \in E}{\arg\max} \quad -\frac{c}{2} \left\| \operatorname{div}_w p - F^k \right\|_2^2 , \tag{67}$$

  where $F^k = {p_s}^k - {p_t}^k + \frac{\lambda^k}{c}$ is fixed.

- Optimize source flow p_s

$$p_s^{k+1} = \underset{p_s(x) \le C_s(x) \ \forall x \in V}{\arg\max} \sum_{x \in V} p_s - \frac{c}{2} \left\| p_s - G^k \right\|_2^2 , \tag{68}$$

  where $G^k = {p_t}^k + \operatorname{div}_w p^{k+1} - \frac{\lambda^k}{c}$ is fixed.

- Optimize sink flow p_t

$$p_t^{k+1} = \underset{p_t(x) \le C_t(x) \ \forall x \in V}{\arg\max} \quad -\frac{c}{2} \left\| p_t - H^k \right\|_2^2 , \tag{69}$$

  where $H^k = {p_s}^{k+1} - \operatorname{div}_w p^{k+1} + \frac{\lambda^k}{c}$ is fixed.

- Update λ

$$\lambda^{k+1} \;=\; \lambda^k - c\,(\operatorname{div}_w p^{k+1} - p_s^{k+1} + p_t^{k+1})\,.$$

To solve (59), construct the augmented Lagrangian function:

$$L_c(p_s, p_t, p, \lambda) = \sum_{x \in V} \left\{ v^b p_s - v^f p_t + \lambda(\operatorname{div}_w p - p_s + p_t) \right\} - \frac{c}{2} \left\| \operatorname{div}_w p - p_s + p_t \right\|_2^2 \,.$$

The goal is to solve

$$\max_{p_s, p_t, p} \sum_{x \in V} (v^b p_s - v^f p_t)$$

under constraints (38) - (41) by solving

$$\min_{\lambda} \max_{p_s, p_t, p, \lambda} L_c(p_s, p_t, p, \lambda).$$

An augmented Lagrangian method can be applied to solve this problem by alternatively maximizing L_c for the dual variables p_s, p_t and p with constraints (60)-(62) and updating the Lagrange multiplier λ. The augmented Lagrangian method for (59) thus becomes:

Convex Method (Version 1s) Supervised Max-flow Algorithm

Initialize p_s^1, p_t^1, p^1 and λ^1. For $k = 1, ...$ until convergence:

- Optimize p flow

$$p^{k+1} = \arg\max_{|p(e)| \leq C \; \forall e \in E} \; -\frac{c}{2} \left\| \operatorname{div}_w p - F^k \right\|_2^2 , \tag{70}$$

where $F^k = {p_s}^k - {p_t}^k + \frac{\lambda^k}{c}$ is fixed.

- Optimize source flow p_s

$$p_s^{k+1} = \arg\max_{p_s(x) \leq 0 \; \forall x \in V} \sum_{x \in V} v^b p_s - \frac{c}{2} \left\| p_s - G^k \right\|_2^2 , \tag{71}$$

where $G^k = {p_t}^k + \operatorname{div}_w p^{k+1} - \frac{\lambda^k}{c}$ is fixed.

- Optimize sink flow p_t

$$p_t^{k+1} = \arg\max_{p_t(x) \leq 0 \; \forall x \in V} \; -\sum_{x \in V} v^f p_t - \frac{c}{2} \left\| p_t - H^k \right\|_2^2 , \tag{72}$$

where $H^k = {p_s}^{k+1} - \operatorname{div}_w p^{k+1} + \frac{\lambda^k}{c}$ is fixed.

- Update λ

$$\lambda^{k+1} \;=\; \lambda^k - c\left(\operatorname{div}_w p^{k+1} - p_s^{k+1} + p_t^{k+1}\right).$$

Due to the relation between problem (64) and problem (33), the output λ at convergence will be a solution to (33). Similarly, if η is chosen sufficiently high in (33), then solution λ to (46) will also be a solution to (33).

By Theorem 1, one can obtain a partition which solves (32) by the thresholding procedure described in (2).

The subproblems (67) and (70) for updating p are solved by one step of the following procedure:

$$p^{k+1} = \Pi_W\left(p^k + c\nabla_w(\text{div}_w p^k - F^k)\right).$$

Above, Π_W is a projection operator which is defined as

$$\Pi_W(p(x,y)) = \begin{cases} p(x,y) & \text{if } |p(x,y)| \le 1, \\ \text{sgn}(p(x,y)) & \text{if } |p(x,y)| > 1, \end{cases} \tag{73}$$

where sgn is the sign function.

The subproblems (68) and (69) can be solved by

$$p_s(x) = \min(G^k(x) + \frac{1}{c}, C_s(x));$$

$$p_t(x) = \min(H^k(x), C_t(x)).$$

The subproblems (71) and (72) can be solved by

$$p_s(x) = \min(G^k(x) + \frac{v^b}{c}, C_s(x));$$

$$p_t(x) = \min(H^k(x) - \frac{v^f}{c}, C_t(x)).$$

Note that the gradient and divergence operators in the algorithm are constructed using the graphical framework, as shown by equations (3) and (4).

4.2 Convex Method (Version 1b): Max-flow *With* Balancing Constraints

This section demonstrates how to incorporate balancing constraints of the form (28), which take into account the size of the classes. This lets the user take advantage of any prior knowledge of the size of the two classes. Volume constraints have been incorporated into image segmentation models in a convex framework in [69, 85]. We propose an efficient algorithm for incorporating the hard volume constraint (28) on graphs by slightly modifying the dual max-flow problem using a new variable $\rho : V \to \mathbb{R}$ as follows:

$$\max_{p_s, p_t, p, \rho} \left\{ P(p_s, p_t, p) \, = \, \sum_{x \in V} (p_s(x) - a\rho) \right\} \tag{74}$$

subject to

$$|p(x,y)| \, \leq \, C, \quad (x,y) \in E; \tag{75}$$

$$p_s(x) \leq C_s(x), \quad \forall x \in V; \tag{76}$$

$$p_t(x) \leq C_t(x), \quad \forall x \in V; \tag{77}$$

$$\operatorname{div}_w p(x) - p_s(x) + p_t(x) + \rho = 0, \quad \forall x \in V; \tag{78}$$

$$\rho \text{ is a constant function.} \tag{79}$$

Introducing a Lagrange multiplier λ for the constraint (78) yields the primal-dual model

$$\min_{\lambda} \max_{p_s, p_t, p} \left\{ E(p_s, p_t, p, \rho; \lambda) = \sum_{x \in V} (p_s(x) - a\rho) + \sum_{x \in V} \lambda(x) \big(\operatorname{div}_w p - p_s + p_t + \rho \big) \right\} \tag{80}$$

subject to

$$|p(x,y)| \, \leq \, C, \quad (x,y) \in E; \tag{81}$$

$$p_s(x) \leq C_s(x), \quad \forall x \in V; \tag{82}$$

$$p_t(x) \leq C_t(x), \quad \forall x \in V; \tag{83}$$

$$\rho \text{ is a constant function.} \tag{84}$$

Rearranging the terms, we obtain

$$\min_{\lambda} \max_{p_s, p_t, p, \rho} \left\{ E(p_s, p_t, p, \rho; \lambda) = \sum_{x \in V} \left\{ \big(1 - \lambda\big) p_s + \lambda p_t + \rho \big(\lambda - a\big) + \lambda \operatorname{div}_w p \right\} \right\} \tag{85}$$

subject to (81) - (77) and (79).

The intuition of having the above model lies in the following. We observe that if $\lambda \notin \mathbb{B}'$, the energy can be arbitrarily large by choosing p_s or p_t arbitrarily small, contradicting boundedness from above. From the second to last term, it follows that if the balancing constraint (28) is not satisfied, the energy can be made arbitrarily large by chosing ρ arbitrarily high or low. Therefore, by maximizing for p_s, p_t, p and ρ, we obtain the closed form expression (33) subject to the constraints (34) and the balancing constraint (28). The derivation is very similar to the one in Section 12.3.1.

In contrast to the case with the model without balancing constraints, it cannot be guaranteed in advance that a global minimizer is obtained by the thresholding procedure described in Theorem 2. If the solution is binary, it must also be a global minimizer of the binary constrained problem, since the convex set $\mathbb{B}'$ contains the binary set $\mathbb{B}$. In the experiments, the solution tends to be binary or very close to binary, indicating that a global minimizer, or close approximation, can be obtained.

We again construct the augmented Lagrangian functional

$$L_c(p_s, p_t, p, \lambda) = \sum_{x \in V} -a\rho + p_s + \lambda(\operatorname{div}_w p - p_s + p_t + \rho) - \frac{c}{2} \left\|\operatorname{div}_w p - p_s + p_t + \rho\right\|_2^2 ,$$

which is exactly (4.1.2) if ρ is zero.

The goal is to solve

$$\max_{p_s, p_t, p, \rho} \left\{ P(p_s, p_t, p) = \sum_{x \in V} (p_s(x) - a\rho) \right\}$$

under constraints (81) - (79) by solving

$$\min_{\lambda} \max_{p_s, p_t, p, \lambda} L_c(p_s, p_t, p, \lambda).$$

An augmented Lagrangian method can be applied to solve this problem by alternatively maximizing L_c for the dual variables p_s, p_t and p with constraints (81)-(79) and updating the Lagrange multiplier λ. The augmented Lagrangian method for (59) thus becomes:

Convex Method (Version 1b) Balancing Constraints

Initialize p_s^1, p_t^1, p^1 and λ^1. For $k = 1, ...$ until convergence:

- Optimize p flow

$$p^{k+1} = \underset{\|p(e)\| \leq C \ \forall e \in E}{\arg\max} \ - \frac{c}{2} \left\|\operatorname{div}_w p - F^k\right\|_2^2 , \tag{86}$$

where $F^k = {p_s}^k - {p_t}^k + \frac{\lambda^k}{c} - \rho^k$ is fixed.

- Optimize source flow p_s

$$p_s^{k+1} = \underset{p_s(x)\leq\, C_s(x)\ \forall x\in V}{\arg\,\max} \sum_{x\in V} p_s\, - \frac{c}{2}\left\|p_s - G^k\right\|_2^2, \tag{87}$$

where $G^k = {p_t}^k + \operatorname{div}_w p^{k+1} - \frac{\lambda^k}{c} + \rho^k$ is fixed.

- Optimize sink flow p_t

$$p_t^{k+1} = \underset{p_t(x)\leq\, C_t(x)\ \forall x\in V}{\arg\,\max} \quad -\frac{c}{2}\left\|p_t - H^k\right\|_2^2, \tag{88}$$

where $H^k = {p_s}^{k+1} - \operatorname{div}_w p^{k+1} + \frac{\lambda^k}{c} - \rho^k$ is fixed.

- Optimize ρ

$$\rho^{k+1} = \arg\max_{\rho} \sum_{x\in V} a\rho - \frac{c}{2}\left\|\rho - J^k\right\|_2^2, \tag{89}$$

where $J^k = -p_t^{k+1} - \operatorname{div}_w p^{k+1} + \frac{\lambda^k}{c} + p_s^{k+1}$ is fixed.

- Update λ

$$\lambda^{k+1} \;=\; \lambda^k - c\,(\operatorname{div}_w p^{k+1} - p_s^{k+1} + p_t^{k+1} + \rho^{k+1})\,.$$

The optimization problem (86) for p can be solved by one step of the projected gradient method as follows:

$$p^{k+1} = \Pi_W(p + c\nabla_w(\operatorname{div}_w p^k - F^k)),$$

where Π_W is the projection defined in (73).

The subproblems (87) and (88) can be solved by

$$p_s(x) = \min(G^k(x) + \frac{1}{c}, C_s(x));$$
$$p_t(x) = \min(H^k(x), C_t(x)).$$

We solve (89) using

$$\rho^{k+1} = \text{mean}(-{p_s}^{k+1} + {p_t}^{k+1} + \operatorname{div}_w p^{k+1} + \rho^k - \frac{\lambda^k}{c} - \frac{a}{c}). \tag{90}$$

In step (90), the constraint that ρ should be constant is imposed exactly by computing the average of the pointwise unconstrained maximizers $\rho(x)$ for $x \in V$, and then the average value is assigned to $\rho(x)\ \forall\ x \in V$.

Just like in the previous cases, the final classification is obtained by thresholding λ.

4.3 Convex Method (Version 2): Extension of Primal Augmented Lagrangian Method to Graphs

In this section, we describe another algorithm for solving the convex problem (32), by extending the Split-Bregman algorithm [55] for geometric problems [54] to general graphs. It has recently been shown [95, 107] that the Split-Bregman algorithm is equivalent to solving a specialized decomposition of total variation regularized problems by the augmented Lagrangian method. We use the augmented Lagrangian notation when describing the algorithm, since this notation has already been introduced in Section 4.1 when deriving the max-flow algorithms.

Consider the general minimization problem

$$\min_{u} \; TV_w(u) + \sum_{x\in V} g(u(x), x). \tag{91}$$

In our case, we wish to choose g according to (31) and impose the constraint $u \in \mathbb{B}'$. The idea is to solve (91) by writing this unconstrained problem as a constrained one by addition an extra variable. This results in the problem

$$\min_{u,q} \|q\|_1 + \sum_{x\in V} g(u(x), x) \tag{92}$$

$$\text{s.t. } q = \nabla_w u, \tag{93}$$

where $\|s\|_1 = \sum_{x\in V} |s(x)|$.

The idea is to solve this by the augmented Lagrangian method. We introduce a Lagrange multiplier ϕ for the constraint (93). This results in the augmented Lagrangian functional

$$L_c(u, q, \phi) = \|q\|_1 + \sum_{x\in V} \big\{ g(u(x), x) + \phi \cdot (q - \nabla_w u) \big\} + \frac{c}{2} \|q - \nabla_w u\|_2^2 , \tag{94}$$

where c is a constant and $\|s\|_2^2 = \sum_{x\in V} |s(x)|^2$.

The idea is to solve (92) with constraint (93) by finding a saddle point of (94) over u, q and ϕ:

$$\max_{\phi} \min_{u,q} L_c(u, q, \phi).$$

This one can achieve by alternating between minimizing for u and q

$$(u^k, q^k) = \arg\min_{u,q} L_c(u, q, \lambda^k) \tag{95}$$

and updating the Lagrange multiplier by one step of gradient ascent:

$$\lambda^{k+1} = \lambda^k + c(q^{k+1} - \nabla u^{k+1}).$$

The minimization problem (95) can be separated into two subproblems:

$$\min_u \sum_{x \in V} \left\{ g(u(x), x) - \lambda^k \cdot \nabla_w u \right\} + \frac{c}{2} \left\| q - \nabla_w u \right\|_2^2 ;$$

$$\min_q \left\| q \right\|_1 + \sum_{x \in V} \lambda^k \cdot q + \frac{c}{2} \left\| q - \nabla_w u \right\|_2^2 .$$

Therefore, the algorithm is the following (it will be referred to in the text as version 2 of the convex method or the primal augmented Lagrangian algorithm):

Convex Method (Version 2) Primal Augmented Lagrangian Algorithm

Initialize λ^1, q^1 and u^1. For $k = 1, ...$ until convergence:

- Optimize λ

$$u^{k+1} = \min_u \sum_{x \in V} \left\{ g(u(x), x) - \lambda^k \cdot \nabla_w u \right\} + \frac{c}{2} \left\| q^k - \nabla_w u \right\|_2^2 . \tag{96}$$

- Optimize q

$$q^{k+1} = \min_q \left\| q \right\|_1 + \sum_{x \in V} u^k \cdot q + \frac{c}{2} \left\| q - \nabla_w u^{k+1} \right\|_2^2 . \tag{97}$$

- Update Lagrange multipliers

$$\lambda^{k+1} = \lambda^k + c(q^{k+1} - \nabla u^{k+1}). \tag{98}$$

The final binary classification is obtained by thresholding u to either 0 or 1.

The subproblem (96) gives the Euler-Lagrange equation:

$$\frac{\partial g}{\partial u} + c \operatorname{div}_w(q^k - \nabla_w u) + \operatorname{div}_w(\lambda^k) = 0,$$

where in this case $\frac{\partial g}{\partial u} = C_t - C_s$.

We solve the above subproblem using one step of forward Euler:

$$\frac{u^{k+1} - u^k}{dt} = -(C_t - C_s + c \operatorname{div}_w(q^k - \nabla_w u^k) + \operatorname{div}_w(\lambda^k)).$$

This becomes

$$\frac{u^{k+1} - u^k}{dt} = -(C_t - C_s + c\,\mathrm{div}_w(q^k) - c\triangle_w u^k) + \mathrm{div}_w(\lambda^k)).$$

All the operators, as was stated before, are formulated in a graph setting, as shown in section 10.

We solve subproblem (97) in the same way as it is done in [107]:

$$q^{k+1} = \begin{cases} \frac{1}{c}(1 - \frac{1}{|b(x,y)|})b(x,y), & \text{if } |b(x,y)| > 1, \\ 0, & \text{if } |b(x,y)| \leq 1, \end{cases}$$

where $b = |c\nabla_w u - \lambda^k|$.

We have tried solving the subproblem (96) above in another way. A similar scheme is used, except the Laplace operator is calculated implicitly, and we proceed further by considering its terms as a linear combination of the eigenvectors of the random walk Laplacian, in a similar way as in [9] and [80]. Only a small fraction of the eigenvectors are used. This way of solving the subproblem turns out to be several times faster than the one previously discussed, but because it does not perform well in the case of non-random fidelity, we did not use it. A disadvantage of this method is that it only uses a small fraction of the eigenvectors, which might not contain enough information to result in an accurate classification, as was the case with experiments with non-random fidelity. However, it also has some advantages, which are discussed in the next section.

Avoiding trivial global minimizers

If the number of supervised points $V^f \cup V^b$ are very low, the global minimizer of (22) may just be the trivial solution $S = V^f$ or $S = V^b$. This was the case for a small number of our experiments. In order to avoid this problem, the cost of these trivial solutions can be increased by increasing the number of edges incident to the supervised points $V^f \cup V^b$, which amounts to adding nonlocal behavior. Non-supervised points in the graph are connected by edges to their M nearest neighbors. Supervised points can instead be connected to their K nearest neighbors, where $K > M$, thereby increasing the cost of the partitions $S = V^f$ and $S = V^b$.

An interesting observation is that if the subproblem (96) in the primal max-flow algorithm is solved via the approximate eigendecomposition, the algorithm does not result in the trivial solution $S = V^f$ and $S = V^b$, even when the number of supervised points are very low. The reason for this seems to be that the approximation resulting from not using all the eigenfunctions erases the unwanted trivial global minimizers from the energy landscape.

Note that the second eigenvector of the Laplacian already provides a solution to a cut using a spectral clustering approximation approach. Although we experiment with using that approximation as an initialization, the methods work just as well when random initialization is used.

4.4 Results

The results for several data sets are summarized in Table 10, with those of the best method highlighted. In all experiments, we have constructed the graph using the M nearest neighbors and approximated the eigendecomposition of the Laplacian using only the M largest eigenvalues. By "random fidelity", we mean choosing supervised points randomly. By "corner fidelity", we mean choosing supervised points in a certain portion of the data set only, in this case in the "corner" portion of the eigenvector graph. The technique described in Section 4.3 for avoiding trivial global minimizers, was used on the two moons data set in Fig.22 in case of less than $3.25\ \%$ supervised points. The results were computed on a 2.4 GHz Intel Core i2 Quad.

In the case when we need to compute the eigenvectors and eigenvalues of the random walk graph Laplacian, we first use a fast numerical solver called the Rayleigh-Chebyshev procedure [1] to compute those of the symmetric graph Laplacian. One can then use the previously described relationship between the eigendecomposition of the symmetric graph Laplacian and that of the random walk graph Laplacian. The Rayleigh-Chebyshev procedure itself is a modification of an inverse subspace iteration method using adaptively determined Chebyshev polynomials. It is also a robust method that converges rapidly and that can handle cases when there are eigenvalues of multiplicity greater than one. The calculations are made even more efficient by the fact that only a small portion of the eigenvectors need to be calculated, as the most significant nodes contain enough information to produce accurate results. To have a fair comparison, we use the same number of eigenvectors per data set for all methods.

We have used

$$f(\lambda(x), x) = \eta(x)|\lambda(x) - \lambda_0(x)|^2 \tag{99}$$

for all our computations. Here, λ_0 is the initial value of λ, and $\eta(x)$ is a function that takes on a value of a constant η on fidelity points and zero elsewhere.

Below, we provide more detail on the results for each of the benchmark data sets, as well as a description of the data set itself. In addition, we provide a comparison of the results to those of some of the best methods, including the binary MBO and GL algorithms. The binary GL algorithm is the multiclass GL algorithm with the number of classes equal to 2.

MNIST Data Set

The MNIST digits data set [72], available at *http://yann.lecun.com/exdb/mnist/*, is a data set of 70000 28×28 images of handwritten digits from $0 - 9$. However, since our method is only binary, we obtained a subset of this set to classify, in particular, digits 4 and 9 (since these digits are sometimes hard to distinguish, if handwritten). This created a set of 13782 digits, each either 4 or 9. Starting from some initial classification of the points and using only a small fraction of the set as fidelity, the goal is to classify each image into being either a 4 or 9. We used $M = 10$.

Using random initialization and random fidelity, the max-flow method obtained an accuracy of 98.48% averaged over 100 runs with different fidelity sets of 500 randomly chosen points (or only 3.62% of the set). The primal augmented Lagrangian method's accuracy was around 98.44%. The accuracy of binary MBO graph method from [80] and the binary GL graph method from [9] was slightly lower than that of our methods; the algorithms were able to achieve an average accuracy of 98.37% and 98.29%, respectively. Table 10 summarizes the above and also shows results in the case when the initialization is constructed using the thresholded second eigenvector of the Laplacian or when the fidelity region is chosen nonrandomly by only considering points that give values in the corners of leftmost image in Figure 24a. More detail on the latter information will be given in Section 4.4. The parameters for Convex Method (Version 1) were: $c = 0.5$, $\eta = 50$. For Convex Method (Version 2), they were: $c = 0.008$, $\eta = 400$, $dt = 0.032$.

To compare with other methods, we note a recent result by Hu et al. [66], which is an unsupervised method. The authors of the paper also tested only digits 4 and 9 and obtained a purity score (measures the fraction of the nodes that have been assigned to the correct community) of 0.977. The GenLouvain algorithm obtained a purity score of 0.975. In addition, many other algorithms have used the full MNIST data set with all 10 digits. For example, Cheeger cuts [100], boosted stumps [68, 72], and transductive classification [101] have obtained accuracies of 88.2%, 92.3%-98.74%, and 92.6%, respectively. Also, papers on k-nearest neighbors [71, 72], neural or convolutional nets [25, 71, 72], nonlinear classifiers [71, 72] and SVM [29, 71] report accuracies of 95.0%-97.17%, 95.3%-99.65%, 96.4%-96.7%, and 98.6%-99.32%, respectively. The aforementioned approaches are supervised learning methods using $60,000$ out of $70,000$ digits (or about

85.71% of the whole data set) as a training set. Morever, we compare our method with [13], which obtains 98.05% accuracy by knowing 10% of the labels, 97.78% by knowing 5% of the labels, and 97.72% by knowing 2.5% of the labels. Our algorithms, taking only 3.6% of the data as fidelity, obtain around 98.5% accuracy, and thus are competitive with or outperform, these methods. Moreover, we have not performed any preprocessing or initial feature extraction on the data set, unlike most of the mentioned algorithms.

Banknote Authentication Data Set

The banknote authentication data set, from the UCI machine learning repository [4], is a data set of 1372 features extracted from images (400×400 pixels) of genuine and forged banknotes. Wavelet transform was used to extract the features from the images. The goal is to segment the banknotes into being either genuine or forged. We used $M = 15$.

The results are shown in Table 10. With the max-flow method, for a 5.1% fidelity set, we were able to obtain an average accuracy (over 100 different fidelity sets) of around 99.09%, while the primal augmented Lagrangian method achieved a similar accuracy of 98.75%. The results did not deteriorate much for a smaller fidelity set of 3.6%, with the two methods achieving an accuracy of 98.83% and 98.29%, respectively. The parameters for Algorithm 1 were: $c = 0.15$, $\eta = 250$. For Algorithm 2, they were: $c = 0.08$, $\eta = 50$, $dt = 0.5$.

We compare this result to the binary MBO algorithm, which achieved a lower accuracy of 95.43% and 93.48% for 5.1% and 3.6% fidelity sets, respectively. For the binary GL method, the results were also not as good- 97.76% and 96.10%, for 5.1% and 3.6% fidelity sets, respectively.

Two Moons Data Set

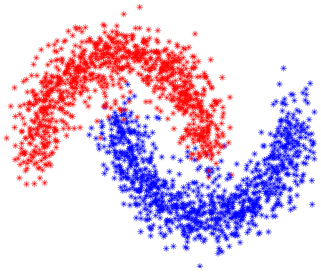


Figure 22: Two moons example with max-flow method

This way the data set is constructed was described in Part 1. We used $M = 10$.

For the max-flow method, in the case of 65 or lower number of fidelity points (3.25 %), we increased the number of edges of supervised points to others to avoid the trivial global minimizer where all points but the supervised ones are classified as one class.

Using random initialization and random fidelity, for the max-flow method, we obtained an average accuracy of 97.10% and 97.05% in the case of 100 and 50 fidelity points, resp.. An example of a solution is shown in Figure 22, with the two classes colored in red and blue. The primal augmented Lagrangian method achieved an accuracy of around 97.07% for 100 fidelity points and around 96.78% for 50 fidelity points. The parameters for Convex Method (Version 1) were: $c = 0.5$, $\eta = 50$. For Version 2, they were: $c = 0.32$, $\eta = 100$, $dt = 0.008$.

To compare this with binary MBO, the method obtained 98.41% and 97.53% accuracy for 100 and 50 fidelity points, resp., which is very similar to the results of the binary GL graph method.

Rods Data Set

We have also tested this algorithm on two other synthetic data sets created using the rods pictured in Figure 23. Around 2000 uniformly chosen points were sampled from each image, and then embedded in $\mathbb{R}^{100}$. Finally, noise was added to each of the points. We used $M = 25$.

In the case of random fidelity region, we obtained accuracy in the 98^{th} or 99^{th} percentile, no matter what initialization. In the case of fidelity region in the corner, we obtained interesting global minimizers for the two data sets. The comparison of our results with the binary MBO and GL method is detailed in the next section. The parameters for Algorithm 1 were: $c = 0.01$, $\eta = 50$. For Algorithm 2, they were: $c = 0.016$, $\eta = 500$, $dt = 0.512$.

Comparison of Convex Method Versions 1 and 1b

We tested our balancing constraints max-flow method on several data sets using random initialization and fidelity, and compared it to the regular max-flow meethod. It handles the case of a small fidelity region better than the original max-flow method (Algorithm 1) and gives higher accuracy everywhere. We used the same M as described in the previous section. The results are displayed in Table 9, and those of the best method (compared to Algorithm 1) are highlighted. In general, the solution is very close to binary, with some small differences that may be explained by the finite stopping criterion of the algorithm, indicating that close approximations to global minimizers are obtained. To measure how close the solution λ is to binary, we have computed the norm

Table 9: Comparison of Convex Method Versions 1 and 1b

Number of fidelity points	50	40	30	20
Two moons- Max-flow method (regular)	97.05%	96.92%	96.86%	88.22%
Two moons- Max-flow method (balancing constraints)	**97.19%**	**97.12%**	**97.11%**	**96.11%**

Number of fidelity points	500	400	300	210
MNIST- Max-flow method (regular)	98.44%	98.40%	98.36%	93.68%
MNIST- Max-flow method (balancing constraints)	**98.59%**	**98.48%**	**98.45%**	**98.41%**

Number of fidelity points	50	40	30	20
Banknote Authentication Data Set (regular)	**98.83%**	98.72%	98.21%	96.78%
Banknote Authentication Data Set (balancing constraints)	**98.83%**	**98.91%**	**98.72%**	**98.55%**

$\sum_{x \in V} \frac{|\lambda(x) - \lambda^{\ell}(x)|}{|V|}$, where λ^{ℓ} is defined in (2) and $\ell = 0.5$. The norm should be 0 if λ is binary. In the experiments, the values of the norm range from 0.0005 to $6 * 10^{-18}$.

For the two moons example, starting from 20 fidelity points, we obtained good results. For 50, 40, 30 and 20 fidelity points, we obtained 97.19%, 97.12%, 97.11% and 96.11% average accuracy (over 100 different fidelity sets), respectively. While the results of the binary MBO method (without any zero means constraint) for the two moons data set achieves slightly better accuracy for 50 and 100 fidelity points (being of 97.53% and 98.41%, respectively), we noticed that if the number of fidelity points is too low, the method is unable to perform well, as the results depending on the fidelity set. The same is the case with the max-flow method with no balancing constraint. For example, for 20 fidelity points, the average accuracy we obtained for the method was 88.22%. However, with the balanced method, we still obtain a good result (96.11%) for a fidelity set containing as little as 20 points. Thus, the advantage of the method is that it performs well with even a very small fidelity region. The results are summarized in Table 9.

For the MNIST data set, we obtained very good results even for a small number of fidelity points. For 500, 400, 300 and 210 fidelity points (or 3.6%, 2.9%, 2.2% and 1.5% of the data, respectively), we obtained an average accuracy (over 100 different fidelity sets) of 98.59%, 98.48%, 98.45% and 98.41%, resp.. A comparison to the results of the regular max-flow method is in Table 9. Note that in addition to giving at least a slightly higher accuracy everywhere, it handles the case

Table 10: Comparison of MBO and Convex Methods

	max-flow	convex method	binary MBO	binary GL
MNIST (3.6% fidelity) random initialization, random fidelity	**98.48%**	98.44%	98.37%	98.29%
MNIST (3.6% fidelity) 2nd eigenvector initialization, random fidelity	**98.48%**	98.43%	98.36%	98.25%
MNIST (3.6% fidelity) random initialization, corner fidelity	**98.47%**	98.40%	62.35%	64.39%
MNIST (3.6% fidelity) 2nd eigenvector initialization, corner fidelity	**98.46%**	98.40%	63.87%	63.19%
Banknote Authentication Data Set (5.1% fidelity)	**99.09%**	98.75%	95.43%	97.76%
Banknote Authentication Data Set (3.6% fidelity)	**98.83%**	98.29%	93.48%	96.10%
two moons (5% fidelity)	97.10%	97.07%	**98.41%**	98.31%
two moons (2.5% fidelity)	97.05%	96.78%	97.53%	**98.15%**

of a small number of fidelity points better than the original method. For example, for 210 fidelity points, the accuracy is 98.41% compared to 93.68% of the original method.

For the banknote authentication data set from the UCI Machine Learning Repository [4], we obtained reasonable results for as little as 20 fidelity points. The results are shown in Table 9. The balancing constraints method obtains better results than the original max-flow method, achieving an average accuracy (over 100 different fidelity sets) of 98.55% for only 20 fidelity points, as opposed to the accuracy of 96.78% of the original max-flow method.

For the first rod data set, we obtained reasonable results starting from around 10 fidelity points out of around two thousand that are in the rods data set. For 10 to 20 fidelity points, the accuracy was around around 96%. Testing 50 and 100 fidelity points, we obtained around 99% accuracy.

Comparison of Our Convex Algorithms to Binary MBO and GL Methods

After comparing the results of our convex algorithms to the binary MBO [80] and binary GL [9] graph methods, we have reached the following conclusions based on our work:

- As long as the fidelity points are well represented for each class (meaning the fidelity points represent a whole variety of points in the class), the binary MBO method and the binary GL method have no trouble finding the correct minimizer or something very close. The

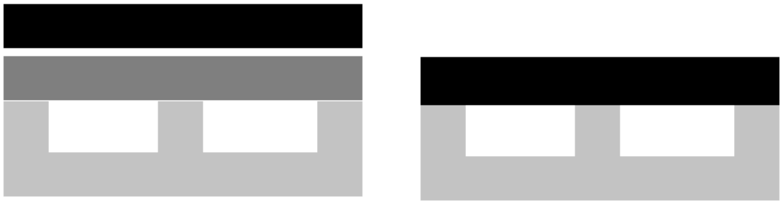


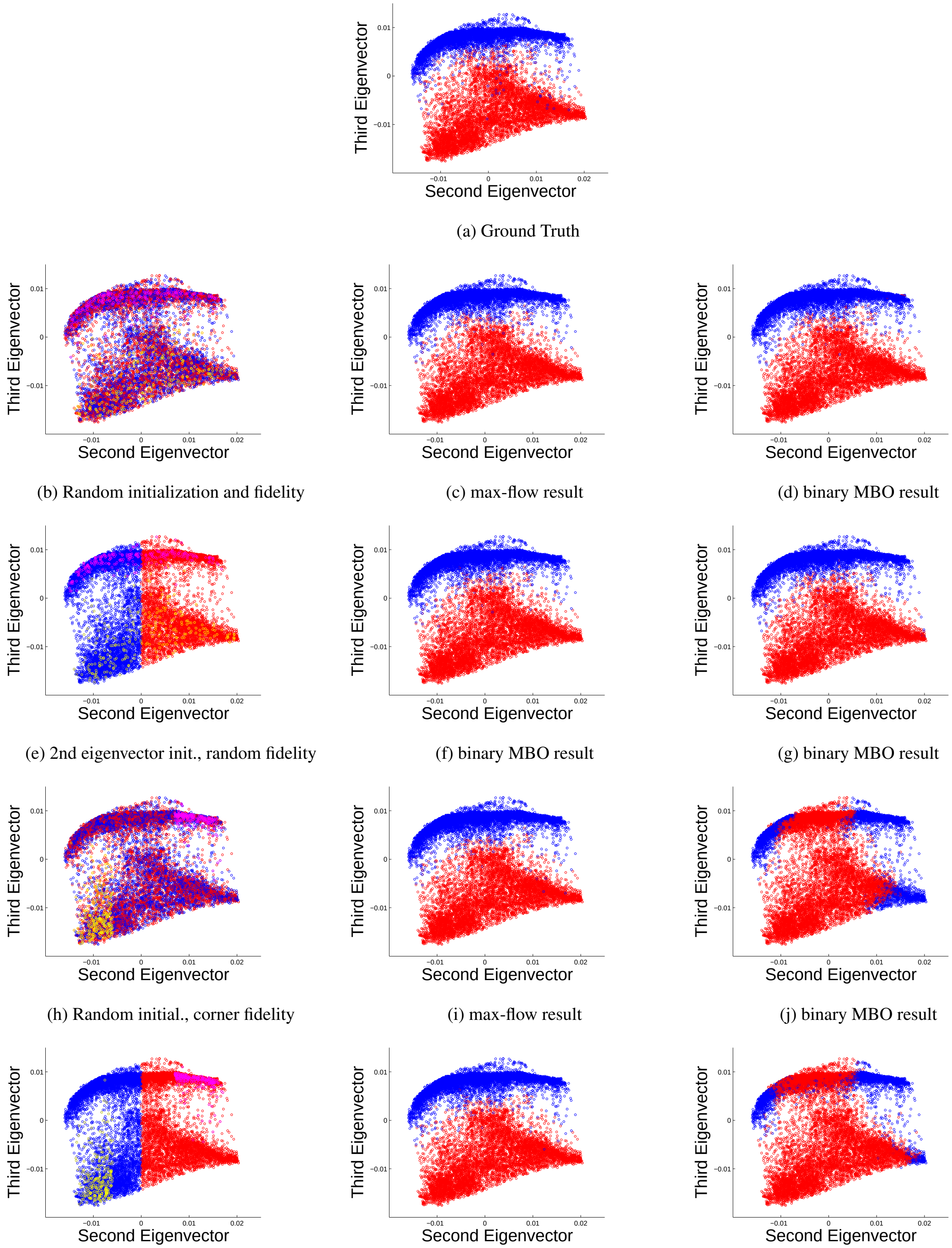
Figure 23: Rod 1 and Rod 2 Data Sets

initialization might not matter; even with a bad initialization, the local minimizer will still be found. Our convex methods find the local minimizer easily.

- Problems occur when the fidelity is not chosen randomly. In this case, even if the initialization is random, the convergence might not occur for the binary MBO and GL methods. In all our experiments with the rods data sets and MNIST, the local minimizer was not found by the two methods. However, our convex algorithms still found the correct local minimizer.

These conclusions are supported by the work done on the MNIST digits data set, using digits 4 and 9 only. The second vs. third eigenvector of the symmetric graph Laplacian are graphed in Figure 24a, with one digit represented by blue and another by red. The results of experiments on this data set are found in Figure 24. Each row represents a different experiment: first two rows contain experiments with random initialization, while last two rows contain experiments with fidelity in a constrained area. The initialization is random for experiments in first and third row, and is constructed by thresholding the second eigenvector of the Laplacian for the results in the second and fourth row. The first column represents the initialization, while the second and third columns are results for the max-flow and binary MBO algorithm, respectively. The fidelity points are marked by yellow and magenta for the two classes.

We see that if the fidelity region is well represented in the data set, no matter what initialization, none of the algorithms have trouble finding a close to perfect solution (accuracy is between 98% and 99%- see Table 10). However, when the fidelity region is not random (in this case constrained to only the nodes whose corresponding entry in the second or third eigenvector of the Laplacian match a certain range), we see that the binary MBO and GL algorithms fail to obtain an accurate solution; its accuracy is below 70%. However, the max-flow algorithm and the primal augmented Lagrangian methods achieve an almost perfect solution of 98.47% and 98.40% accuracy, resp.



(a) Ground Truth

(b) Random initialization and fidelity

(c) max-flow result

(d) binary MBO result

(e) 2nd eigenvector init., random fidelity

(f) binary MBO result

(g) binary MBO result

(h) Random initial., corner fidelity

(i) max-flow result

(j) binary MBO result

(k) 2nd eigenvector init., corner fidelity

(l) max-flow result

(m) binary MBO result

Figure 24: MNIST Data set Results. Left: initialization, supervised points are marked in yellow and magenta. Middle: max-flow algorithm result. Right: binary MBO result

The two conclusions are also supported by the work done on the two rod data sets created from images displayed in Figure 23. Experiments with the first and second rod image are shown in Figures 25 and 26, respectively. The first two rows represent cases with a random fidelity set, while the last two rows are experiments with corner fidelity. Cases with random initialization are in first and third rows, and second eigenvector initialization is used in experiments in the second and fourth row. The first column is the initialization, second column is the max-flow algorithm result, and third column is the binary MBO result.

For rod 1, we found that when the fidelity is chosen randomly, the minimizer divides the bottom two rods from the rest of the image. When the fidelity is chosen at the corners, the minimizer is shown in the bottom two rows of the second column. The convex max-flow and primal augmented Lagrangian algorithms are able to attain these minimizers, while the binary MBO and GL algorithms struggle in the case of non-random fidelity. The situation is similar with rod 2.

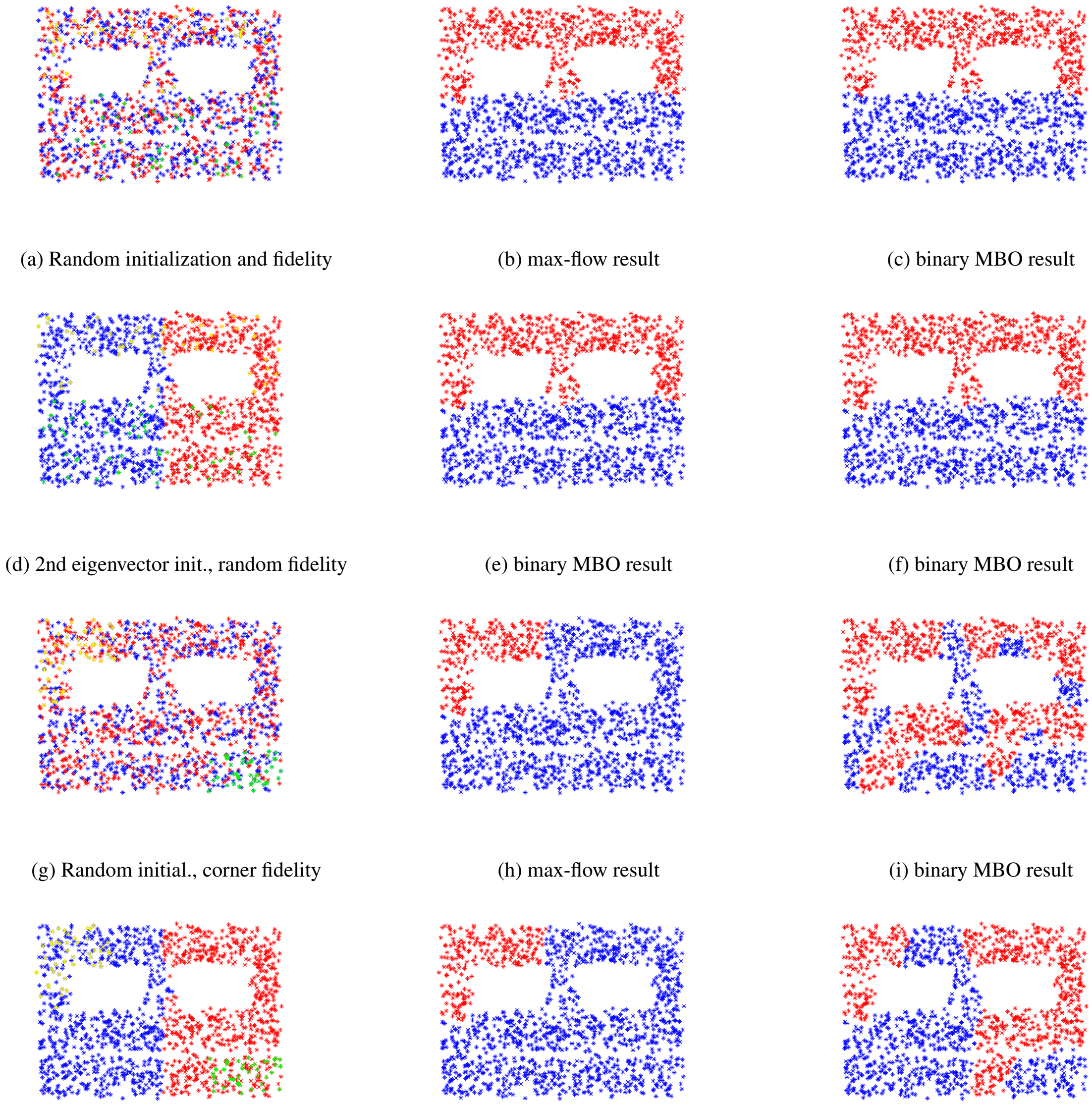
Note about MBO Algorithm

As noted in [80], the first step of the MBO algorithm (heat equation with an extra term) was solved using the eigenvalue and eigenvector decomposition of the Laplacian. However, a disadvantage of solving equations using this method is that, for it to be efficient, only a fraction of eigenvectors are used, and they might not contain enough information to result in an accurate classification. Naturally, the more eigenvectors one computes, the longer the process will take.

As an alternative way of solving the first step (30) of the MBO algorithm, we have tried using just the simple forward heat equation solver. However, this did not result in an accurate segmentation in the case of non-random fidelity, thus not improving the results from the original way of solving it. This shows that the algorithm is getting stuck in a local minimum, since the problem is clearly not the lack of information encoded within the small number of eigenvectors used.

Comparison of Convergence, Speed, and Energy

The stopping criterion used for all algorithms was taken to be the point at which the square of the relative L2 norm between the current and previous iterate is negligible, or below a certain constant α. With the exception of the MNIST data set (where $\alpha = 5 * 1e - 10$), the max-flow, binary MBO and GL algorithms stabilize around $\alpha = 1e - 17$ or $1e - 16$. The primal augmented Lagrangian method stabilizes around $\alpha = 1e - 08$ or $1e - 09$.



(a) Random initialization and fidelity

(b) max-flow result

(c) binary MBO result

(d) 2nd eigenvector init., random fidelity

(e) binary MBO result

(f) binary MBO result

(g) Random initial., corner fidelity

(h) max-flow result

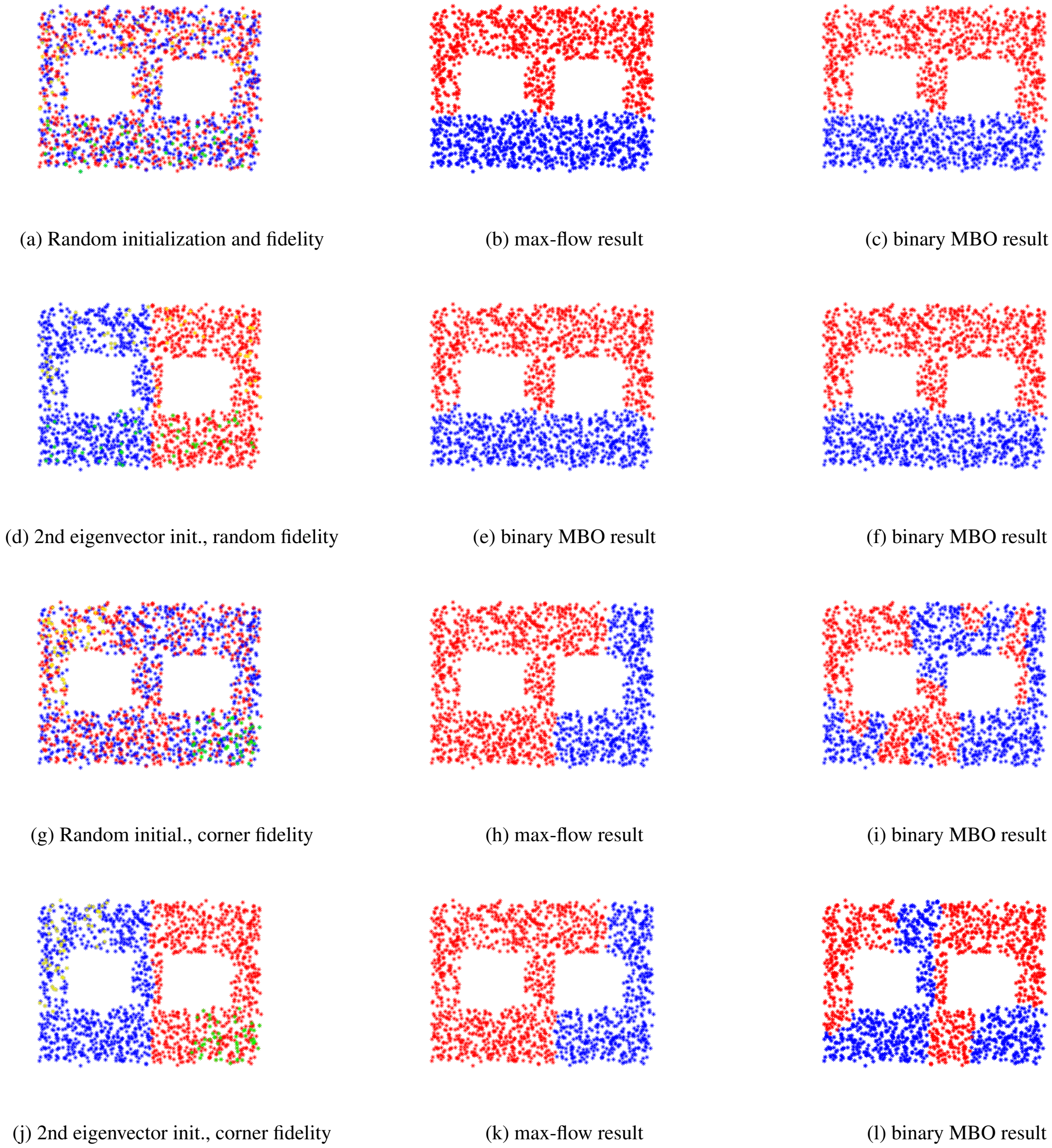
(i) binary MBO result

(j) 2nd eigenvector init., corner fidelity

(k) max-flow result

(l) binary MBO result

Figure 25: Rod 1 Data Set Results. Left: initialization, supervised points are marked in yellow and green. Middle: max-flow algorithm result. Right: binary MBO result



(a) Random initialization and fidelity

(b) max-flow result

(c) binary MBO result

(d) 2nd eigenvector init., random fidelity

(e) binary MBO result

(f) binary MBO result

(g) Random initial., corner fidelity

(h) max-flow result

(i) binary MBO result

(j) 2nd eigenvector init., corner fidelity

(k) max-flow result

(l) binary MBO result

Figure 26: Rod 2 Data Set Results. Left: initialization, supervised points are marked in yellow and green. Middle: max-flow algorithm result. Right: binary MBO result

Table 11: Number of Iterations of MBO and Convex Methods and Runtime Comparison

Number of iterations	max-flow	primal augmented Lagrangian	binary MBO	binary GL
MNIST	426	2709	10	52
Banknote Authentication Data Set	314	725	7	449
two moons	1031	451	8	108

Timing (s)	max-flow	primal augmented Lagrangian	binary MBO	binary GL
MNIST [a]	2.88	43.21	0.52	0.78
Banknote Authentication Data Set	1.21	3.76	0.90	0.95
two moons	4.13	5.23	2.30	2.98

[a] This is the timing of the method using already computed weights and eigenvalues/eigenvectors of the random walk Laplacian.

Table 12: Comparison of Final Energy of the MBO and Convex Methods

Data Set	initial energy	max-flow final energy	primal augmented Lagrangian final energy	binary MBO final energy	binary GL final energy
MNIST (random fid)	23223	789	789	798	804
MNIST (non-random fid)	23223	791	792	2167	5363
Banknote Authentication	3308	30	37	51	42
two moons	3802	533	535	538	548
rod 1 (random fid)	4159	146	148	163	159
rod 1 (non-random fid)	4159	88	89	825	391
rod 2 (random fid)	4528	171	176	186	184
rod 2 (non-random fid)	4528	101	105	709	421

Table 11 includes information about the number of iterations needed to reach stability, and also the timing results for each data set.

We have also computed the initial and final energy for each data set. The energy was calculated using

$$E(\lambda) = \frac{1}{2} \sum_{x,y \in V} w(x,y)|u(x) - u(y)|, \tag{100}$$

where $\lambda(x)$ is 0 if node x was classified to be in the first class, and 1 if it was classified to be in the second class. Note that the energy here is exactly $TV_w(\lambda)$. Table 12 includes information about the initial and final energy for each method. We see that the max-flow algorithm is able to obtain the lowest energy in each case. In general, one can see that the convex algorithms are able to obtain the global minimizer in all cases, while the binary MBO and GL algorithms struggle in the case of non-random fidelity.

CHAPTER 5

Modified Cheeger Method

5.1 Derivation of the Method

We consider the binary partitioning problem of the target set X. The ratio cut problem can be written as minimizing

$$cut(S, \bar{S})\Big(\frac{1}{|S|} + \frac{1}{|\bar{S}|}\Big)$$

over all sets S, where S is a subset of X. We have $X = S \cup \bar{S}$.

Now let u be a binary function (taking values 0 or 1) denoting the class of each of the nodes. Then one can write the ratio cut problem as the minimization of

$$\min_{u:u(x)\in\{0,1\}} \big(\frac{1}{2}\sum_{x,y} w(x,y)|u(x) - u(y)|\big)\Big(\frac{1}{\sum_x u(x)} + \frac{1}{n - \sum_x u(x)}\Big)$$

over all binary functions (with values 0 or 1).

Similar logic is used in [12] which states that the problem of minimizing

$$\frac{cut(S, \bar{S})}{min(|S|, |\bar{S}|)}$$

over all partitions $X = S \cup \bar{S}$ reduces to minimizing

$$\frac{\frac{1}{2}\sum_{x,y} w(x,y)|u(x) - u(y)|}{\sum_x |u(x) - med(u)|}$$

over all binary functions u (with values 0 or 1), where $med(u)$ is the median of u.

Using the same notation, one can write the following formulas for the total variation and Ginzburg-Landau (GL) functionals. These definitions were used in previous papers by Yves van Gennip et al. in [103]:

$$TV_w(u) = \frac{1}{2}\sum_{x,y} w(x,y)|u(x) - u(y)|,$$

$$GL_\epsilon(u) = \frac{1}{2}\sum_{x,y} w(x,y)(u(x) - u(y))^2 + \frac{1}{\epsilon}\sum_x W(u(x)).$$

The justification of why there is no ϵ in the first term of Ginzburg-Landau function is justified in [103]. Also, note that the cut can be exactly interpreted as total variation. In other words, minimizing the cut is the same as minimizing the total variation of the binary function (indicating the class) that takes values 0 or 1 on the nodes.

Assume n is the number of vertices in the graph and let $\mathcal{V} \cong \mathbb{R}^n$ and $\mathcal{E} \cong \mathbb{R}^{\frac{n(n-1)}{2}}$ be Hilbert spaces defined via the following inner products:

$$\langle \lambda, \gamma \rangle_{\mathcal{V}} = \sum_x \lambda(x)\gamma(x)d(x),$$

$$\langle \psi, \phi \rangle_{\mathcal{E}} = \frac{1}{2}\sum_{x,y} \psi(x,y)\phi(x,y)w(x,y).$$

Here, d_i represents the degree of node i.

Let us also define the following norms:

$$||\lambda||_{\mathcal{V}} = \sqrt{\langle \lambda, \lambda \rangle_{\mathcal{V}}} = \sqrt{\sum_x \lambda(x)^2 d(x)},$$

$$||\phi||_{\mathcal{E}} = \sqrt{\langle \phi, \phi \rangle_{\mathcal{E}}} = \sqrt{\frac{1}{2}\sum_{x,y} \phi(x,y)^2 w(x,y)}.$$

Then, one can rewrite the GL functional using the inner product notation:

$$GL_\epsilon(u) = ||\nabla u||^2_{\mathcal{E}} + \frac{1}{\epsilon}\langle D^{-r}W(u), 1\rangle_{\mathcal{V}},$$

where D is the degree diagonal matrix and $(D^{-1}W(u))(x) = d(x)^{-1}W(u_i)$. The factor $d(x)^{-1}$ is needed to cancel the factor $d(x)$ in the $\mathcal{V}$- inner product.

The work [103] has the following theorem:

Theorem 3. *$GL_\epsilon \xrightarrow{\Gamma} GL_0$ as $\epsilon \to 0$ where*

$$GL_0(\lambda) = \begin{cases} TV_w(u) & \textit{for } u \textit{ s.t. } u_i \in \{0,1\}, \\ \infty & \textit{otherwise}. \end{cases}$$

Therefore, one can interchange total variation (a.k.a. cut) with the Ginzburg-Landau functional. Replacing the TV term (the first term) in (5.1) by the GL functional, we obtain the problem of minimizing

$$F(u) = GL_\epsilon(u)\Big(\frac{1}{\sum_x u(x)} + \frac{1}{n - \sum_x u(x)}\Big).$$

We denote the second term in the product by $C_1(u)$.

The modified Allen-Cahn equation can then be derived using the $\mathcal{V}$-gradient flow associated with GL functional. Since

$$\frac{d}{dt}F(u+tv) = (\frac{d}{dt}GL_\epsilon(u+tv))(C_1(u+tv)) + GL_\epsilon(u+tv)(\frac{d}{dt}C_1(u+tv)),$$

evaluation at zero gives

$$-2\langle C_1(u)\triangle u, v\rangle_V + \frac{1}{\epsilon}\langle C_1(u)D^{-1}W'(u), v\rangle_V + \langle GL(u)D^{-1}C_2(u), v\rangle_V,$$

where D is the diagonal matrix containing the degree of the node in its diagonal places,

$$C_1(u) = \frac{1}{\sum_x u(x)} + \frac{1}{n - \sum_x u(x)}$$

and

$$C_2(u) = \frac{-1}{(\sum_x u(x))^2} + \frac{1}{(n - \sum_x u(x))^2}.$$

The modified Allen-Cahn equation is thus the following:

$$\frac{\partial u(x)}{\partial t} = 2C_1(u)(\triangle u)(x) - \frac{C_1(u)W'(u(x))}{\epsilon d(x)} - \frac{GL_\epsilon(u)C_2(u)}{d(x)}$$

or equivalently

$$\frac{\partial u(x)}{\partial t} = 2C_1(u)(\triangle u)(x) - \frac{C_2(u)||\nabla u||^2}{d(x)} - \frac{C_1(u)W'(u(x))}{\epsilon d(x)} - \frac{(\sum_i W(u(x)))C_2(u)}{\epsilon d(x)}, \quad (101)$$

where $d(x)$ is the degree of node x.

5.2 The Algorithm

Discretizing (101), calculating the Laplacian implicitly, one obtains

$$\frac{u_i^{n+1} - u_i^n}{dt} = 2C_1(u^{n+1})(\triangle u^{n+1})_i - \frac{C_2(u^n)||\nabla u^n||^2}{d_i} - \frac{C_1(u^n)W'(u_i^n)}{\epsilon d_i} - \frac{(\sum_i W(u_i^n))C_2(u^n)}{\epsilon d_i}. \quad (102)$$

To solve the above equation, we use the eigendecomposition of a Laplacian. Let

$$u^n = \sum_k a_k^n \phi_k(x)$$

$$-\frac{C_2(u)||\nabla u||^2}{d_i} - \frac{C_1(u)W'(u_i)}{\epsilon d_i} - \frac{(\sum_i W(u_i))C_2(u)}{\epsilon d_i}) = \sum_k b_k^n \phi_k(x),$$

where $\phi(x)$ are the eigenfunctions of a Laplacian. After plugging in these repsentations into (102), one obtains

$$a_k^{n+1} = \frac{a_k^n + dt b_k^n}{1 + 2dt C_1(u^n)\lambda_k},$$

where λ_k is the k^{th} eigenvalue of a Laplacian.

Therefore, the algorithm consists of the following five steps:

- Initialize u^0.
- Repeat the following steps from $n = 0$ until a stopping criterion is satisfied:
  - Calculate $C_1 = C_1(u^n)$, $C_2 = C_2(u^n)$.
  - Calculate a^{n+1} using (5.2).
  - Calculate b^{n+1} using the transform of u^n.
  - Calculate u^{n+1} using the inverse transform of a^{n+1}.
- Threshold the solution to obtain a binary answer.

One can derive a similar algorithm using a slightly different problem of minimizing

$$cut(S, \bar{S})\Big(\frac{1}{vol(S)} + \frac{1}{vol(\bar{S})}\Big)$$

over all sets S, where S is a subset of X. We have $X = S \cup \bar{S}$. Now, C_1 and C_2 would be the following:

$$C_1(u) = \frac{2}{\sum_x d(x)u(x)} + \frac{2}{\sum_x d(x)(1-u(x))},$$
$$C_2(u) = \frac{-2}{(\sum_x d(x)u(x))^2} + \frac{2}{\sum_x d(x)(1-u(x)))^2}.$$

In addition, (102) will be changed to a similar one but in which the normalization by the d_i from the second and fourth terms in the right hand side of the equation.

However, the results of the two algorithms do not differ by much, and thus we only state the results for the original algorithm in the results section.

Moreover, we are considering developing an MBO scheme for this problem, in a similar way as was done in Chapter 3. However, this is not trivial due to the difficulties encountered with the new thresholding step because of the presence of the normalization factor in the problem formulation.

5.3 General Laplacian Framework

Similar to the procedure in [50], we proceed with testing different kinds of Laplacians. To derive some new Laplacians, we focus on the case of a dynamic process. In a dynamic process, each node θ_i is associated with a dynamic variable which is allowed to change. The variables evolve according to a dynamic process. In [50], the process considered was

$$\frac{d\Theta}{dt} = -\mathcal{L}\Theta,$$

where Θ is a matrix containing the θ_i entries. The matrix $\mathcal{L}$ is called the spreading operator.

We consider the more general framework for the Laplacian:

$$\mathcal{L} = T^{-\frac{1}{2}}D^{-\frac{1}{2}}(D - Z)D^{-\frac{1}{2}}T^{-\frac{1}{2}}.$$

In this representation, T is an $n \times n$ diagonal matrix of nodel delay factors. It's i^{th} diagonal element represents the average delay of node i. We also consider a generalization Z of the traditional weight matrix. Z can be any symmetric positive- definite matrix. We want to study how the results change when Z and T vary. We consider four cases:

Normalized Laplacian In the case when $Z = W$ and $T = I$, we obtain the normalized symmetric Laplacian:

$$\mathcal{L}_{\textbf{sym}} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}.$$

Scaled Graph Laplacian In the case when $Z = W$, $T = d_{max}D^{-1}$ (here, d_{max} represents the maximum degree in the degree matrix associated with Z), the spreading operator is called the scaled graph Laplacian:

$$\mathcal{L}_{\textbf{scl}} = \frac{1}{d_{max}}(D - W).$$

Replicator If we let $Z = VWV$, where V is a diagonal matrix whose elements are the components of the eigenvector corresponding to the largest eigenvalue of W, and $T = I$, we obtain

$$\mathcal{L}_{\textbf{rep}} = I - \frac{1}{\lambda_{max}}W,$$

where λ_{max} is the largest eigenvalue of W.

Unbiased Adjacency Matrix If we let $Z = D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$ and $T = d_{max}D^{-1}$ (here, d_{max} represents the maximum degree in the degree matrix associated with Z), we obtain the unbiased adjacency matrix:

$$\mathcal{L}_{\textbf{unb}} = \frac{1}{d_{max}}(D - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}).$$

Table 13: Modified Cheeger Method Results

Data Set	symmetric Laplacian	scaled graph Laplacian	replicator	unbiased adjacency matrix
MNIST	98.27%	98.12%	94.38%	98.26%
two moons	97.8%	97.45%	94.85%	97.55%

Table 14: Modified Cheeger Method Results and Comparison

MNIST

Method	Accuracy
symmetric Laplacian	98.27%
scaled graph Laplacian	98.12%
replicator	94.34%
unbiased adjacency matrix	98.26%
method in [12]	98.36 %

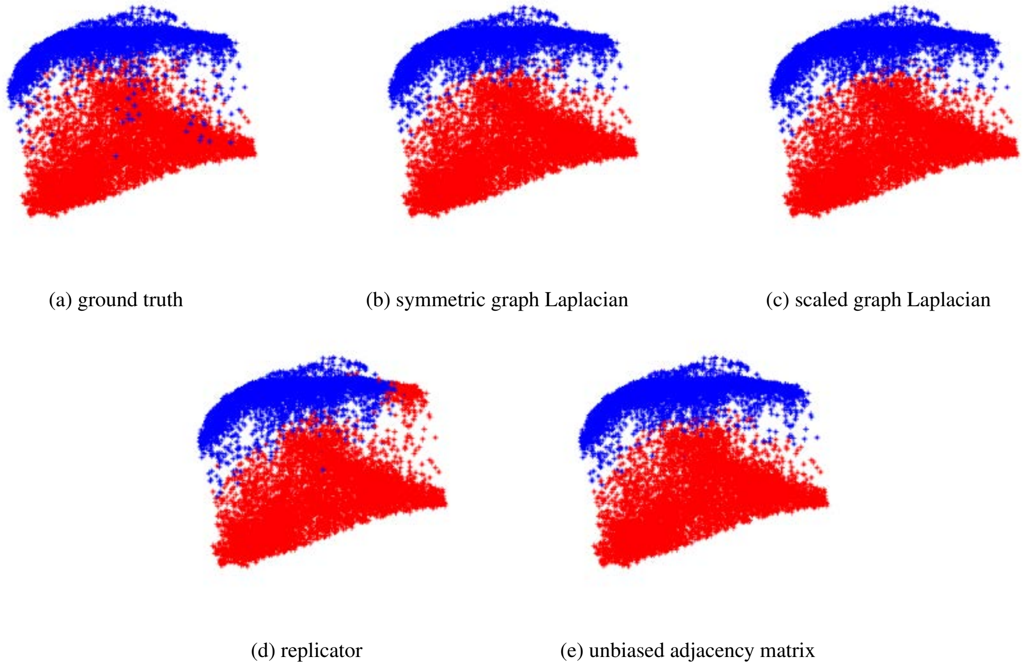
Two moons

Method	Accuracy
symmetric Laplacian	97.8%
scaled graph Laplacian	97.45%
replicator	94.85%
unbiased adjacency matrix	97.55%
method in [100]	95.08 %
method in [18]	93.5 %
method in [63]	95.38 %
method in [14]	91.31%
method in [12]	95.86 %

5.4 Results

MNIST Data Set

The MNIST data set results are shown in Table 13. We test all four Laplacians shown in the previous section. We see that all but the replicator achieve accuracy in the 98^{th} percentile. Using the replicator worsens the accuracy to 94^{th} percentile. To compare to some state-of-the-art work involving some kind of a cut (i.e. Cheeger cut, balanced cut, etc.) or spectral computations, we note the result of 98.36% of Bresson et al. in [12]. We see our method achieves a result that is very similar.



(a) ground truth

(b) symmetric graph Laplacian

(c) scaled graph Laplacian

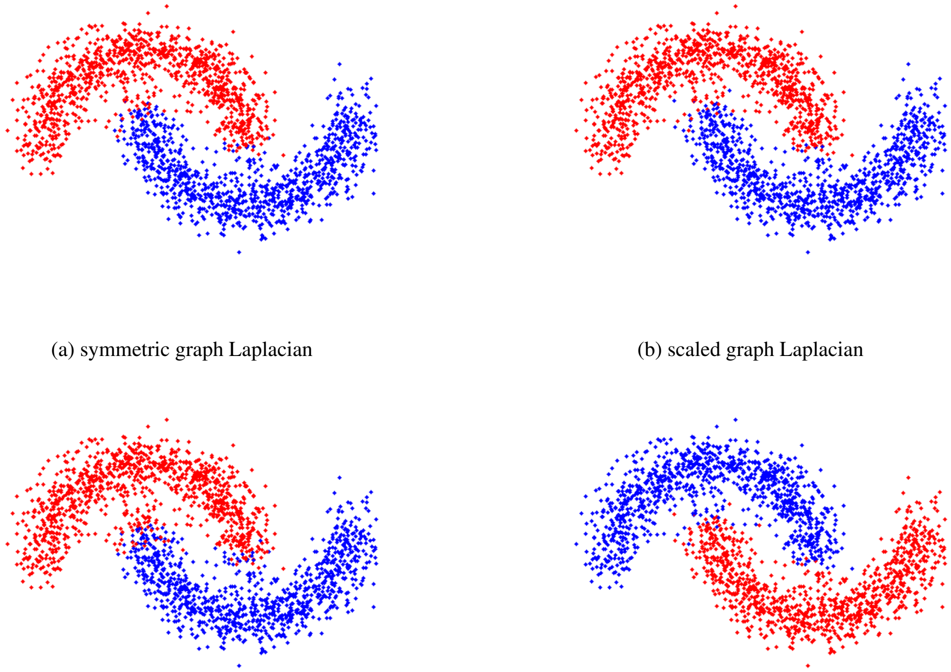
(d) replicator

(e) unbiased adjacency matrix

Figure 27: Modified Cheeger Method MNIST Data Set Results

Two Moons Data Set

The two moons data set results are shown in Table 13. We test all four Laplacians shown in the previous section. We see that all but the replicator achieve accuracy in the 97^{th} percentile. Using the replicator worsens the accuracy to 94^{th} percentile. To compare to some state-of-the-art work involving some kind of a cut (i.e. Cheeger cut, balanced cut, etc.) or spectral computations, we note the 95.08% result of Szlam et al in [100], the 93.5% result of Buhler et al in [18], the 95.38% result of Hein et al in [63], the 91.31% result of Bresson et al in [14] and the 95.86% result of Bresson et al in [12]. We see that our method achieves an accuracy that is at least 2% higher.



(a) symmetric graph Laplacian

(b) scaled graph Laplacian

(c) replicator

(d) unbiased adjacency matrix

Figure 28: Modified Cheeger Method Two Moons Data Set Results

Summary

We have introduced several methods using a graphical setting. The advantage of using a graphical setting is that it is a more general framework which also allows one to use nonlocal information. In images, it allows one to capture texture and repetitive structure.

First, an efficient semi-supervised algorithm (MBO method) for classification and image processing was described. The main advantage of the method lies in its efficiency and simplicity, since it consists of just alternating between solving a modified heat equation and thresholding. The multiclass extension of the algorithm was presented using the Gibbs simplex. The method can be applied to many areas, and an application to hyperspectral imagery was also described. Only around 2%-5% of the data is needed to be known to obtain an accurate answer.

One drawback of the MBO algorithm is that it is formulated from a non-convex optimization problem, so it may be the case that the result is a local minimum. To overcome this problem, we present two versions of solving a convex optimization problem. The first one is an equivalent maximum-flow problem, which is solved using the augmented Lagrangian method. We also discuss some variations of the version, such as solving it using balancing constraints and hard supervised constraints. The second version is solving the problem in its original form using an extra variable and then proceeding with the augmented Lagrangian method. This results in two efficient algorithms that can be applied to the problem of binary classification/segmentation. The extension to the multiclass class is a problem of future research. Just like the MBO method, these versions are semi-supervised, although only a very small percentage of the data is needed to be known to achieve accurate results.

We also present a modified Cheeger method for the problem of binary segmentation/classfication. Similarly to the MBO method, the root of this algorithm also involves the Ginzburg-Landau functional. The advantage of this method is that it is unsupervised, and thus requires no prior knowledge, unlike the previous mentioned methods.

Appendix A

Our methods involve the computation of eigenvalues and associated eigenvectors of the symmetric graph Laplacian. In practice, one computes only a fraction of the eigenvalues and eigenvectors, and different methods of doing so are used depending on the size of the domain.

When the graph is sparse and is of moderate size, around 5000×5000 or less, we use a Rayleigh-Chebyshev procedure outlined in [1]. It is a modification of an inverse subspace iteration method that uses adaptively determined Chebyshev polynomials. The procedure is also a robust method that converges rapidly and that can handle cases when there are eigenvalues of multiplicity greater than one.

When the graph is very large, such as in the case of image segmentation, the Nyström extension method is used.

Nyström extension for fully connected graphs

Nyström extension [44,45] is a matrix completion method often used in many image processing applications, such as kernel principle component analysis [34] and spectral clustering [87]. This procedure performs much faster than many alternate techniques because it uses approximations based on calculations on small submatrices of the original large matrix. When the size of the matrix becomes very large, this method is especially valuable.

Note that if λ is an eigenvalue of $\hat{W} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, then $1 - \lambda$ is an eigenvalue of L_s, and the two matrices have the same eigenvectors. We formulate a method to calculate the eigenvectors and eigenvalues of $\hat{W}$ and thus of L_s.

Let w be the similarity function, λ be an eigenvalue of W, and ϕ its associated eigenvector. The Nyström method approximates the eigenvalue equation

$$\int_{\Omega} w(y,x)\phi(x)dx = \lambda\phi(x)$$

using a quadrature rule, a technique to find weights $c_j(y)$ and a set of L interpolation points $X = \{x_j\}$ such that

$$\sum_{j=1}^{L} c_j(y)\phi(x_j) = \int_{\Omega} w(y,x)\phi(x)dx + E(y),$$

where $E(y)$ represents the error in the approximation.

We use $c_j(y) = w(y, x_j)$ and choose the L interpolation points randomly from the vertex set V. Denote the set of L randomly chosen points by $X = \{x_i\}_{i=1}^{L}$ and its complement by Y. Partioning Z into $Z = X \cup Y$ and letting $\phi_k(x)$ be the the k^{th} eigenvector of W and λ_k its associated eigenvalue, we obtain the system of equations

$$\sum_{x_j \in X} w(y_i, x_j)\phi_k(x_j) = \lambda_k \phi_k(y_i) \qquad \forall y_i \in Y, \ \ \forall k \in 1, ..., L.$$

This system of equations cannot be solved directly since the eigenvectors are not known. To overcome this problem, the L eigenvectors of W are approximated using calculations involving submatrices of W.

Let W_{XY} be defined as

$$\begin{bmatrix} w(x_1, y_1) & \dots & w(x_1, y_{N-L}) \\ \vdots & \ddots & \vdots \\ w(x_L, y_1) & \dots & w(x_L, y_{N-L}) \end{bmatrix},$$

where W has dimension $N \times N$. The matrices W_{YX}, W_{XX} and W_{YY} can be defined similarly. Notice that W_{XY}= ${W_{YX}}^T$. Then the matrix W can be written as

$$\begin{bmatrix} W_{XX} & W_{XY} \\ W_{YX} & W_{YY} \end{bmatrix}.$$

To calculate the eigenvalues and eigenvalues of $\hat{W}$, one must correctly normalize the above weight matrix. The correct normalization is achieved by the following calculations, where we denote by $\mathbf{1}_K$ the K-dimensional unit vector.

Let the matrices d_X and d_Y be defined as

$$d_X = W_{XX}\mathbf{1}_L + W_{XY}\mathbf{1}_{N-L},$$
$$d_Y = W_{YX}\mathbf{1}_L + (W_{YX}W_{XX}^{-1}W_{XY})\mathbf{1}_{N-L}.$$

If $A./B$ denotes componentwise division between matrices A and B, and v^T denotes the transpose of vector v, then define the matrices $\hat{W}_{XX}$ and $\hat{W}_{XY}$ as

$$\begin{aligned}\hat{W}_{XX} &= W_{XX}./(s_X s_X^T),\\ \hat{W}_{XY} &= W_{XY}./(s_X s_X^Y),\end{aligned}$$

where $s_X = \sqrt{d_X}$ and $s_Y = \sqrt{d_Y}$.

It is shown in [9] that if $\hat{W}_{XX} = B_X D B_X^T$, and if A and Γ are matrices such that

$$A^T \Gamma A = \hat{W}_{XX} + \hat{W}_{XX}^{-\frac{1}{2}} \hat{W}_{XY} \hat{W}_{YX} \hat{W}_{XX}^{-\frac{1}{2}}$$

then the eigenvector matrix V consisting of L eigenvectors of $\hat{W}$ and thus of L_s is given by

$$\left[\begin{array}{c} B_X D^{\frac{1}{2}} B_X^T A \Gamma^{-\frac{1}{2}} \\ \hat{W}_{YX} B_X D^{-\frac{1}{2}} B_X^T A \Gamma^{-\frac{1}{2}} \end{array} \right],$$

while $I - \Gamma$ contains the corresponding eigenvalues of L_s in its diagonal entries.

Therefore, the efficiency of the Nyström extension method lies with the fact that when computing the eigenvalues and eigenvectors of an $N \times N$ matrix, where N is large, it approximates them using calculations involving only much smaller matrices, the largest of which has dimension $N \times L$, where L is small.

Although this method is very efficient, there are problems when it is applied to binary image inpainting, especially when the image has a repetitive structure. This occurs because of singular or nearly singular matrices that arise in the calculations of the Nyström extension method. Therefore, in this case, we use the Rayleigh-Chebyshev procedure of [1] to calculate the eigenvalues and associated eigenvectors.

References

[1] C. Anderson. A Rayleigh-Chebyshev procedure for finding the smallest eigenvalues and associated eigenvectors of large sparse Hermitian matrices. *Journal of Computational Physics*, 229:7477–7487, 2010.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J.L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. *Proceedings of International Workshop of Ambient Assisted Living*, pages 216–223, 2012.

[3] P. Arias, V. Caselles, and G. Sapiro. A variational framework for non-local image inpainting. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 345–358, 2009.

[4] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[5] G. Barles and C. Georgelin. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal on Numerical Analysis*, 32(2):484–500, 1995.

[6] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

[7] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003.

[8] A. Bertozzi, S. Esedoḡlu, and A. Gillette. Inpainting of binary images using the Cahn-Hilliard equation. *IEEE Transactions on Image Processing*, 16(1):285–291, 2007.

[9] A.L. Bertozzi and A. Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10(3):1090–1118, 2012.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222 –1239, 2001.

[11] X. Bresson, S. Esedoglu, P. Vandergheynst, J.P. Thiran, and S. Osher. Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and Vision*, 28(2):151–167, 2007.

[12] X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht. Convergence and energy landscape for Cheeger cut clustering. *Advances in Neural Information Processing Systems*, 25:1394–1402, 2012.

[13] X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht. Multiclass total variation clustering. *Advances in Neural Information Processing Systems*, pages 1421–1429, 2013.

[14] X. Bresson, T. Laurent, D. Uminsky, and J.H. von Brecht. An adaptive total variation algorithm for computing the balanced cut of a graph. *arXiv preprint arXiv:1302.2717*, 2013.

[15] X. Bresson, X.-C. Tai, T.F. Chan, and A. Szlam. Multi-class transductive learning based on ; 1 relaxations of cheeger cut and mumford-shah-potts model. *Journal of Mathematical Imaging and Vision*, 49(1):191–201, 2014.

[16] J.B. Broadwater, D. Limsui, and A.K. Carr. A primer for chemical plume detection using LWIR sensors. Technical report, National Security Technology Department, 2011.

[17] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[18] T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. *International Conference on Machine Learning*, pages 81–88, 2009.

[19] J.-F. Cai, R.H. Chan, and Z. Shen. A framelet-based image inpainting algorithm. *Applied and Computational Harmonic Analysis*, 24(2):131–149, 2008.

[20] A. Cardoso. Datasets for single-label text categorization.

[21] T. F. Chan, S. Esedoḡlu, and M. Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal of Applied Mathematics*, 66(5):1632–1648, 2006.

[22] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*, volume 2. MIT Press, 2006.

[23] Y. Chen and X. Ye. Projection onto a simplex. *arXiv preprint arXiv:1101.6081*, 2011.

[24] F.R.K. Chung. *Spectral graph theory*, volume 92. American Mathematical Society, 1997.

[25] D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. *International Joint Conference on Artificial Intelligence*, pages 1237–1242, 2011.

[26] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watershed: a unifying graph-based optimization framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1384–1399, 2011.

[27] C. Couprie, L. Grady, H. Talbot, and L. Najman. Combinatorial continuous maximum flow. *SIAM Journal on Imaging Sciences*, 4(3):905–930, 2011.

[28] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. *Conference on Artificial Intelligence*, pages 509–516, 1998.

[29] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46(1):161–190, 2002.

[30] X. Desquesnes, A. Elmoataz, and O. Lezoray. Pdes level sets on weighted graphs. *IEEE International Conference on Image Processing*, pages 3377–3380, 2011.

[31] T.G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Researh*, 2:263–286, 1995.

[32] J.A. Dobrosotskaya and A.L. Bertozzi. A wavelet-laplace variational technique for image deconvolution and inpainting. *IEEE Transactions on Image Processing*, 17(5):657–663, 2008.

[33] J.A. Dobrosotskaya and A.L. Bertozzi. Wavelet analogue of the Ginzburg-Landau energy and its Γ-convergence. *Interfaces and Free Boundaries*, 12(2):497–525, 2010.

[34] P. Drineas and M.W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[35] I. Ekeland and R. Téman. *Convex analysis and variational problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[36] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: a framework for image and manifold processing. *IEEE Transactions on Image Processing*, 17(7):1047–1060, 2008.

[37] S. Esedoḡlu, S. Ruuth, and R. Tsai. Threshold dynamics for high order geometric motions. *Interfaces and Free Boundaries*, 10(3):263–282, 2008.

[38] S. Esedoḡlu, S. Ruuth, and R. Tsai. Diffusion generated motion using signed distance functions. *Journal of Computational Physics*, 229(4):1017–1042, 2010.

[39] S. Esedoḡlu and Y.H.R. Tsai. Threshold dynamics for the piecewise constant Mumford–Shah functional. *Journal of Computational Physics*, 211(1):367–384, 2006.

[40] L.C. Evans. Convergence of an algorithm for mean curvature motion. *Indiana University Mathematics Journal*, 42(2):533–557, 1993.

[41] D.J. Eyre. An unconditionally stable one-step scheme for gradient systems, 1998.

[42] G. Facciolo, P. Arias, V. Caselles, and G. Sapiro. Exemplar-based interpolation of sparsely sampled images. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 331–344, 2009.

[43] L.R. Ford and D.R. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[44] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

[45] C. Fowlkes, S. Belongie, and J. Malik. Efficient spatiotemporal grouping using the Nyström method. *Computer Society Conference on Computer Vision and Pattern Recognition*, 1:I–231, 2001.

[46] C. Frohn-Schauf, S. Henn, and K. Witsch. Nonlinear multigrid methods for total variation image denoising. *Computing and Visualization in Science*, 7(3-4):199–206, 2004.

[47] C. Garcia-Cardona, A. Flenner, and A.G. Percus. Multiclass diffuse interface models for semi-supervised learning on graphs. *International Conference on Pattern Recognition Applications and Methods*, 2013.

[48] C. Garcia-Cardona, E. Merkurjev, A.L. Bertozzi, A. Flenner, and A. Percus. Fast multiclass segmentation using diffuse interface methods on graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1600–1613, 2014.

[49] T. Gerhart, J. Sunu, L. Lieu, E. Merkurjev, J.-M. Chang, J. Gilles, and A.L. Bertozzi. Detection and tracking of gas plumes in LWIR hyperspectral video sequence data. *SPIE Conference on Defense Security and Sensing*, 2013.

[50] R. Ghosh, S.-H. Teng, K. Lerman, and X. Yan. The interplay between dynamics and networks: centrality, communities, and cheeger inequality. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1406–1415, 2014.

[51] G. Gilboa and S. Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Modeling & Simulation*, 6(2):595–630, 2007.

[52] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2008.

[53] E Giusti. *Minimal surfaces and functions of bounded variation*. Springer Science & Business Media, 1984.

[54] T. Goldstein, X. Bresson, and S. Osher. Geometric applications of the split Bregman method: segmentation and surface reconstruction. *Journal of Scientific Computing*, 45(1):271–293, 2010.

[55] T. Goldstein and S. Osher. The split Bregman method for L_1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[56] L. Grady. Multilabel random walker image segmentation using prior models. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 763–770, 2005.

[57] L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

[58] L. Grady and J.R. Polimeni. *Discrete calculus: applied analysis on graphs for computational science*. 2010.

[59] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. *Visualization, Imaging and Image Processing*, pages 423–429, 2005.

[60] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of computational physics*, 49(3):357–393, 1983.

[61] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

[62] M. Hein, J. Audibert, and U. Von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. *Conference on Learning Theory*, pages 470–485, 2005.

[63] M. Hein and T. Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse PCA. *Advances in Neural Information Processing Systems*, 23:847–855, 2010.

[64] M. Hein and S. Setzer. Beyond spectral clustering - tight relaxations of balanced graph cuts. *Advances in Neural Information Processing Systems*, pages 2366–2374, 2011.

[65] M. Hinnrichs, J. O. Jensen, and G. McAnally. Handheld hyperspectral imager for standoff detection of chemical and biological aerosols. *Optical Technologies for Industrial, Environmental, and Biological Sensing*, pages 67–78, 2004.

[66] H. Hu, T. Laurent, M.A. Porter, and A.L. Bertozzi. A method based on total variation for network modularity optimization using the MBO scheme. *SIAM Journal of Applied Mathematics*, 73(6):2224–2246, 2013.

[67] T. Joachims. Transductive learning via spectral graph partitioning. *International Conference on Machine Learning*, 20(1):290–297, 2003.

[68] B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. *International Conference on Machine Learning*, pages 497–504, 2009.

[69] M. Klodt and D. Cremers. A convex framework for image segmentation with moment constraints. *IEEE International Conference on Computer Vision*, pages 2236 – 2243, 2011.

[70] R.V. Kohn and P. Sternberg. Local minimizers and singular perturbations. *Proceedings of Royal Society of Edinburgh Section A*, 111(1-2):69–84, 1989.

[71] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[72] Y. LeCun and C. Cortes. The MNIST database of handwritten digits.

[73] J. Lellmann, J. H. Kappes, J. Yuan, F. Becker, and C. Schnörr. Convex multi-class image labeling by simplex-constrained total variation. Technical report, IWR, University of Heidelberg, 2008.

[74] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1699 –1712, 2008.

[75] O. Lezoray, A. Elmoataz, and V.-T. Ta. Nonlocal pdes on graphs for active contours models with applications to image segmentation and data clustering. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 873–876, 2012.

[76] Y. Li and J. Kim. Multiphase image segmentation using a phase-field model. *Computers & Mathematics with Applications*, 62(2):737–745, 2011.

[77] Y. Lou, X. Zhang, S. Osher, and A.L. Bertozzi. Image recovery via nonlocal operators. *Journal of Scientific Computing*, 42(2):185–197, 2010.

[78] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008.

[79] D. Manolakis, C. Siracusa, and G. Shaw. Adaptive matched subspace detectors for hyperspectral imaging applications. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 5:3153–3156, 2001.

[80] E. Merkurjev, T. Kostic, and A.L. Bertozzi. An MBO scheme on graphs for classification and image processing. *SIAM Journal on Imaging Sciences*, 6(4):1903–1930, 2013.

[81] B. Merriman, J.K. Bence, and S. Osher. Diffusion generated motion by mean curvature. *AMS Selected Lectures in Mathematics Series: Computational Crystal Growers Workshop*, 8966:73–83, 1992.

[82] B. Merriman, J.K. Bence, and S.J. Osher. Motion of multiple functions: a level set approach. *Journal of Computational Physics*, 112(2):334–363, 1994.

[83] B. Merriman and S.J. Ruuth. Diffusion generated motion of curves on surfaces. *Journal of Computational Physics*, 225(2):2267–2282, 2007.

[84] B. Mohar. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics and Applications*, 2:871–898, 1991.

[85] T. Mollenhoff, C. Nieuwenhuis, and D. Toppe, E.and Cremers. Efficient convex optimization for minimal partition problems with volume constraints. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 94–107, 2013.

[86] Y. Mroueh, T. Poggio, L. Rosasco, and Jean-Jacques E. Slotine. Multi-class learning: with simplex coding. 2011.

[87] M.M. Naeini, G. Dutton, K. Rothley, and G. Mori. Action recognition of insects using spectral clustering. *IAPR Conference on Machine Vision Applications*, pages 1–4, 2007.

[88] S.A. Nene, S.K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). *Technical Report CUCS-006-96*, 1996.

[89] S. Osher and J. Shen. Digitized PDE method for data restoration. *Analytic-Computational Methods in Applied Mathematics*, 698, 2000.

[90] P. Perona and L. Zelnik-Manor. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17:1601–1608, 2004.

[91] G. Peyré, S. Bougleux, and L.D. Cohen. Non-local regularization of inverse problems. *Inverse Problems and Imaging*, 5(2):511–530, 2011.

[92] J. Rubinstein, P. Sternberg, and J.B. Keller. A simple proof of convergence for an approximation scheme for computing motions by mean curvature. *SIAM Journal of Applied Mathematics*, 49(1):116–133, 1989.

[93] S.J. Ruuth. Efficient algorithms for diffusion-generated motion by mean curvature. *Journal of Computational Physics*, 144(2):603–625, 1998.

[94] H. Schaeffer and S. Osher. A low patch-rank interpretation of texture. *SIAM Journal on Imaging Sciences*, 6(1):226–262, 2013.

[95] S. Setzer. Operator splittings, Bregman methods and frame shrinkage in image processing. *International Journal of Computer Vision*, 92(3):265–280, 2011.

[96] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[97] B. Simons. *Phase transitions and collective phenomena.* University of Cambridge, 1997.

[98] A. Subramanya and J. Bilmes. Semi-supervised learning with measure propagation. *Journal of Machine Learning Research*, 12:3311–3370, 2011.

[99] J. Sunu, J.M. Chang, and A.L. Bertozzi. Simultaneous spectral analysis of multiple video sequence data for LWIR gas plumes. *SPIE Conference on Defense, Security, and Sensing*, 2014.

[100] A. Szlam and X. Bresson. A total variation-based graph clustering algorithm for Cheeger ratio cuts. *International Conference on Machine Learning*, pages 1039–1046, 2010.

[101] A. Szlam, M. Maggioni, and R.R. Coifman. Regularization on graphs with function-adapted diffusion processes. *Journal of Machine Learning Research*, 9:1711–1739, 2008.

[102] G. Tochon, J. Chanussot, J. Gilles, M. Dalla Mura, J.-M. Chang, and A.L. Bertozzi. Gas plume detection and tracking in hyperspectral video sequences using binary partition trees. *IEEE Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, 2014.

[103] Y. van Gennip and A.L. Bertozzi. Gamma-convergence of graph Ginzburg-Landau functionals. *Advanced in Differential Equations*, 17(11–12):1115–1180, 2012.

[104] Y. van Gennip, N. Guillen, B. Osting, and A.L. Bertozzi. Mean curvature, threshold dynamics, and phase field theory on finite graphs. *Milan Journal of Mathematics*, 82(1):3–65, 2014.

[105] D. Wagner and F. Wagner. *Between min cut and graph bisection.* Springer, 1993.

[106] J. Wang, T. Jebara, and S.F. Chang. Graph transduction via alternating minimization. *International Conference on Machine Learning*, pages 1144–1151, 2008.

[107] C. Wu and X.-C. Tai. Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models. *SIAM J. Imaging Sci.*, 3(3):300–339, 2010.

[108] J. Yuan, E. Bae, and X.-C. Tai. A study on continuous max-flow and min-cut approaches. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2217–2224, 2010.

[109] J. Yuan, E. Bae, X.-C. Tai, and Y. Boykov. A spatially continuous max-flow and min-cut framework for binary labeling problems. *Numerische Mathematik*, 126(3):559–587, 2013.

[110] A.L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). *Neural Computation*, 15(4):915–936, 2003.

[111] T.F. Zhang, X.and Chan. Wavelet inpainting by nonlocal total variation. *Inverse Problems and Imaging*, 4(1):191–210, 2010.

[112] X. Zhang, M. Burger, X. Bresson, and S. Osher. Bregmanized nonlocal regularization for deconvolution and sparse reconstruction. *SIAM Journal on Imaging Sciences*, 3(3):253–276, 2010.

[113] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

[114] D. Zhou and B. Schölkopf. A regularization framework for learning from graph data. *International Conference on Machine Learning*, 2004.

[115] X. Zhu. Semi-supervised learning literature survey. *Computer Sciences Technical Report 1530, University of Wisconsin-Madison*, 2005.